Software and Systems Modeling manuscript No.
(will be inserted by the editor)

What Makes a Good Process Model?

Lessons Learned From Process Mining

W.M.P. van der Aalst

Eindhoven University of Technology, The Netherlands
e-mail: w.m.p.v.d.aalst@tue.nl

Received: date / Revised version: date

Abstract There seems to be a never ending stream of new process modeling notations. Some of these notations are foundational and have been around for decades (e.g., Petri nets). Other notations are vendor specific, incremental, or are only popular for a short while. Discussions on the various competing notations concealed the more important question *"What Makes a Good Process Model?"*. Fortunately, large scale experiences with process mining allow us to address this question. Process mining techniques can be used to extract knowledge from event data, discover models, align logs and models, measure conformance, diagnose bottlenecks, and predict future events. Today's processes leave many trails in data bases, audit trails, message logs, transaction logs, etc. Therefore, it makes sense to relate these event data to process models independent of their particular notation. Process models discovered based on the actual behavior tend to be very different from the process models made by humans. Moreover, conformance checking techniques often reveal important deviations between models and reality. The lessons that can be learned from process mining shed a new light on process model quality. This paper discusses the role of process models and lists *seven problems* related to process modeling. Based on our experiences in over 100 process mining projects, we discuss these problems. Moreover, we show that these problems can be addressed by exposing process models and modelers to event data.

Key words process mining – process modeling – process model quality

1 Introduction

Today there exists a wide variety of process modeling notations. Some notations have been around for decades. Other notations have been proposed

more recently. Notations range from languages aiming to provide a formal basis (e.g., finite state machines, Petri nets, and process algebras) to vendor specific notations (e.g., the different workflow languages used by BPM vendors). Industry standards such as BPEL [10] and BPMN [18] are typically only partially adopted; vendors support just subsets of these standards and users tend to use only a tiny fraction of the concepts offered [17]. Obviously, there is little consensus on the modeling language to be used. This resulted in the "Tower of Babel of Process Languages": a plethora of similar but subtly different languages inhibiting effective and unified process support and analysis. This "Tower of Babel" and the corresponding discussions obfuscated more foundational questions related to the *correspondence of a model and reality* and the *usefulness and quality of process models*. Therefore, this paper aims to answer the question: *"What Makes a Good Process Model?"*. It would be presumptuous to suggest that this paper provides a conclusive answer to this question. We merely report on our experiences with process mining thereby stimulating the reader to view the topic from a new angle.

Process mining is an emerging research area combining techniques from process modeling, model-based analysis, data mining, and machine learning. The goal is to extract knowledge about processes from event data stored in databases, transaction logs, message logs, etc. Process mining techniques are commonly classified into: (a) *discovery*, (b) *conformance*, and (c) *enhancement* [3]. 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. An example is the α-algorithm [3,8] that is able to discover a Petri net based on sequences of events. 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 [3,21]. The third type of process mining is *enhancement*. Here, the idea is to extend or improve an existing process model using information about the actual process recorded in some event log [3]. 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. One type of enhancement is *repair*, i.e., modifying the model to better reflect reality. Another type of enhancement is *extension*, i.e., adding a new perspective to the process model by cross-correlating it with the log. An example is the extension of a process model with performance data. For instance, by using timestamps in the event log it is possible to identify bottlenecks in a process.

Over the last decade, process mining emerged as a new discipline [3, 15]. Recently, the *IEEE Task Force on Process Mining* released the *Process Mining Manifesto* [15]. This manifesto is supported by 53 organizations and 77 process mining experts contributed to it. The active participation from end-users, tool vendors, consultants, analysts, and researchers illustrate the growing significance of process mining as a bridge between data mining and business process modeling. In fact, there is a growing number of commercially available software products offering process mining capabilities. Some

of these products embed process mining functionality in a larger system, e.g., Pallas Athena embeds process mining in their BPM suite BPM|one. Other products aim at simplifying process mining using an intuitive user interface (Disco, Reflect, Interstage BPME, ProcessAnalyzer, ARIS PPM, etc.). Next to these commercial systems, there is the open-source ProM framework (`www.processmining.org`). ProM has become the de facto standard for process mining in the scientific community and ideas developed in the context of ProM have been adopted in commercial systems. In terms of process mining functionality, the scope of ProM – with its hundreds of plug-ins developed by research groups all over the globe – is much broader than any other "business intelligence" tool.

We have applied ProM in over 100 organizations, e.g., in the context of joint research projects, Master projects, and consultancy projects. This provided us with a refreshing view on process models:

- Process mining allows for the *discovery* of process models *based on facts*. Often the real process has surprising "features" that the stakeholders are not aware of (e.g., exceptional paths or skipped activities).
- Given a pre-existing model and an event log, conformance analysis can be used to detect *differences between reality and model*. Often, such analysis immediately reveals that the process model provides an oversimplified or simply incorrect view on reality. Such diagnostics can be used to align models and reality, thus improving the accuracy and usefulness of models.
- Process mining establishes an explicit connection between elements in the process model and events observed in reality. This way it is possible to *breathe life into otherwise static models*. As a result, process models will not end up in some archive without anyone looking at them. Connecting models and event logs will stimulate the active/daily use of process models. However, such use of process models also imposes new requirements.
- The automatic generation of process models and the projection of event logs and running cases on these models, shows that it does not make any sense to aim at a creating a single process model. For the process of interest, one would like to *generate different models depending on the questions one would like to answer*. Process models should be like geographic maps; there may be different maps covering the same area (hiking map, cycling map, highway map, city map, etc.).

In this paper, we report on the main lessons that can be learned from process mining. In Section 2, we briefly introduce process mining. Based on this, we discuss the role of process models (Section 3). Section 4 discusses seven typical problems related to process modeling. These summarize generic lessons learned through our experiences with process mining. To conclude, Section 5 discusses how event logs can breathe life into otherwise static process models.

2 Process Mining: An Example

Starting point for process mining is a so-called *event log*. An event log contains information on *process instances*, often referred to as *cases*. Each case is described by a sequence of events also called *trace*. Let us start with the simple case where each event is completely described by the corresponding activity name. Table 1 shows a small event log taken from [3]. In such an abstract description of an event log, each case is represented by a sequence of activities. For clarity the activity names have been transformed into single-letter labels, e.g., a denotes activity *register request*.

Table 1 Compact representation of an event log just showing the activity names per event: a = *register request*, b = *examine thoroughly*, c = *examine casually*, d = *check ticket*, e = *decide*, f = *reinitiate request*, g = *pay compensation*, and h = *reject request*

case id	trace
1	$\langle a, b, d, e, h\rangle$
2	$\langle a, d, c, e, g\rangle$
3	$\langle a, c, d, e, f, b, d, e, g\rangle$
4	$\langle a, d, b, e, h\rangle$
5	$\langle a, c, d, e, f, d, c, e, f, c, d, e, h\rangle$
6	$\langle a, c, d, e, g\rangle$
...	...

Process mining algorithms for process discovery can transform the information shown in Table 1 into process models. For instance, the basic α-algorithm [8] discovers the Petri net shown in Fig. 1. It is easy to check that all six traces in Table 1 are possible in the model. Let us replay the trace of the first case – $\langle a, b, d, e, h\rangle$ – to show that the trace "fits" (i.e., conforms to) the model. In the initial marking shown in Fig. 1, transition a is indeed enabled because of the token in *start*. Recall that a transition in a Petri net is *enabled* when each input place contains a token, i.e., all input places are marked. Enabled transitions can *fire* thereby removing a token from each input place and producing a token for each output place. After firing the enabled transition a the places *c1* and *c2* are marked, i.e., both places contain a token. Transition b is enabled at this marking and its execution results in the marking with tokens in *c2* and *c3*. Now we have executed $\langle a, b\rangle$ and the sequence $\langle d, e, h\rangle$ remains. The transition corresponding to the next event in the trace, i.e. d, is indeed enabled and its execution results in the marking enabling e (tokens in places *c3* and *c4*). Firing e results in the marking with one token in *c5*. This marking enables the final event h in the trace. After executing h, the case ends in the desired final marking with just a token in place *end*. Similarly, it can be checked that the

other five traces shown in Table 1 are also possible in the model and that all of these traces result in the marking with just a token in place *end*.

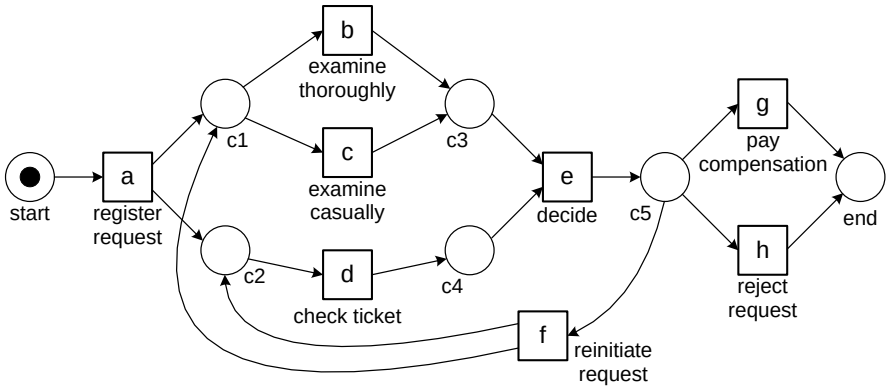


Fig. 1 The process model discovered by the α-algorithm [8] based on the event log in Table 1

The Petri net shown in Fig. 1 also allows for traces not present in Table 1. For example, the traces $\langle a, d, c, e, f, b, d, e, g\rangle$ and $\langle a, c, d, e, f, c, d, e, f, c, d, e, f, c, d, e, f, b, d, e, g\rangle$ are also possible. This is a desired phenomenon as the goal is *not* to represent just the *particular set of example traces* in the event log. Process mining techniques look for the most likely model explaining the underlying process. Therefore, process discovery algorithms need to generalize the behavior contained in the log to show the most likely underlying model that is not invalidated by the next set of observations. One of the challenges of process mining is to balance between "overfitting" (the model is too specific and only allows for the "accidental behavior" observed) and "underfitting" (the model is too general and allows for behavior unrelated to the behavior observed).

The α-algorithm is just one of many possible process discovery algorithms. For real-life logs more advanced algorithms are needed to better balance between "overfitting" and "underfitting" and to deal with "incompleteness" (i.e., logs containing only a small fraction of the possible behavior due to the large number of alternatives) and "noise" (i.e., a log containing exceptional/infrequent behavior that should not automatically be incorporated in the model).

Process mining is not limited to process discovery. Event logs can be used to check conformance and enhance existing models. Moreover, different perspectives may be taken into account. To illustrate this, let us first consider the event log shown in Table 2. The first six cases are as before. It is easy to see that Case 7 with trace $\langle a, b, e, g\rangle$ is not possible according to the model in Fig. 1. The model requires the execution of d before e, but d did not occur. This means that the ticket was not checked at all before making

Table 2 Another event log: cases 7, 8, and 10 are not possible according to Fig. 1

case id	trace
1	$\langle a, b, d, e, h\rangle$
2	$\langle a, d, c, e, g\rangle$
3	$\langle a, c, d, e, f, b, d, e, g\rangle$
4	$\langle a, d, b, e, h\rangle$
5	$\langle a, c, d, e, f, d, c, e, f, c, d, e, h\rangle$
6	$\langle a, c, d, e, g\rangle$
7	$\langle \mathbf{a}, \mathbf{b}, \mathbf{e}, \mathbf{g}\rangle$
8	$\langle \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{g}\rangle$
9	$\langle a, d, c, e, f, d, c, e, f, b, d, e, h\rangle$
10	$\langle \mathbf{a}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}, \mathbf{c}, \mathbf{b}, \mathbf{d}, \mathbf{g}\rangle$

a decision and paying compensation. Conformance checking techniques aim at discovering such discrepancies [4, 21]. A so-called *alignment* showing the problem is:

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|}\hline a & b & \gg & e & g \\ \hline a & b & d & e & g \\ \hline \end{array}$$

The top row of an alignment corresponds to "moves in the log" and the bottom row corresponds to "moves in the model". If a move in the log cannot be mimicked by a move in the model, then a "$\gg$" ("no move") appears in the bottom row. If a move in the model cannot be mimicked by a move in the log, then a "$\gg$" ("no move") appears in the top row. For example, in γ_1 the log did not do a d move whereas the model has to make this move to enable e. Given a trace in the event log there may be many possible alignments. The goal is to find the alignment with the least number of $\gg$ elements. The number of $\gg$ elements can be used to quantify fitness [4]. When checking the conformance of the remainder of the event log it can also be noted that cases 8 and 10 do not conform either. Case 9 conforms although it is not identical to one of the earlier traces. Trace $\langle a, b, c, d, e, g\rangle$ (i.e. Case 8) has the problem that two examinations (b and c) took place whereas the model allows for only one. Alignment γ_2 shows that the second examination in the log cannot be mimicked by a move in the model:

$$\gamma_2 = \begin{array}{|c|c|c|c|c|c|}\hline a & b & c & d & e & g \\ \hline a & b & \gg & d & e & g \\ \hline \end{array}$$

Trace $\langle a, c, d, e, f, c, b, d, g\rangle$ (Case 10) has two problems as shown by alignment γ_3.

$$\gamma_3 = \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline a & c & d & e & f & c & b & d & g \\ \hline a & c & d & e & f & c & \gg & d & g \\ \hline \end{array}$$

Note that 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*, i.e., capture or

predict reality. The second viewpoint is taken when the model is *normative*, i.e., used to influence or control reality.

Table 1 provides a rather simplistic view on event logs. Most event logs also contain information on resources, timestamps and characteristics of cases. For example, the first event of Case 1 corresponds to the execution of activity *register request*. The real event log may show that this activity was executed by Pete on December 30th 2010. The log may also contain transactional information showing that the activity was started at noon and took 30 minutes. Characteristics of the request may include customer type, claimed amount, and execution costs. Depending on the type of event, there can be various attributes. Such information can be used to discover other perspectives, check the conformance of models that are not pure control-flow models, and to extend models with additional information. For example, one could derive a social network based on the interaction patterns between individuals. The social network can be based on the "handover of work" metric, i.e., the more frequent individual x performed an activity that is causally followed by an activity performed by individual y, the stronger the relation between x and y is [7].

Figure 2 illustrates the way in which a control-flow oriented model can be extended. Process mining may reveal that Sara is the only one performing the activities *decide* and *reinitiate* request. This suggests that there is a "manager role" and that Sara is the only one having this role. Activity *examine thoroughly* is performed only by Sue and Sean. This suggests some "expert role" associated to this activity. The remaining activities are performed by Pete, Mike and Ellen. This suggests some "assistant role" as shown in Fig. 2. Techniques for organizational process mining [22] can discover such organizational structures and relate activities to resources through roles. By exploiting resource information in the log, the organizational perspective can be added to the process model. Similarly, information on timestamps and frequencies can be used to add performance related information to the model. Figure 2 sketches that it is possible to measure the time that passes between an examination (activity b or c) and the actual decision (activity e). If this time is remarkably long, process mining can be used to identify the problem and discover possible causes. If the event log contains case-related information, this can be used to further analyze the decision points in the process. For instance, through decision-point analysis it may be learned that requests for compensation of more than € 800 tend to be rejected.

Using process mining, the different perspectives can be cross-correlated to find surprising insights. Examples of such findings could be: "requests examined by Sean tend to be rejected more frequently", "requests for which the ticket is checked after examination tend to take much longer", "requests of less than € 500 tend to be completed without any additional iterations". Moreover, these perspectives can also be linked to conformance questions. For example, it may be shown that Pete is involved in relatively many incorrectly handled requests.

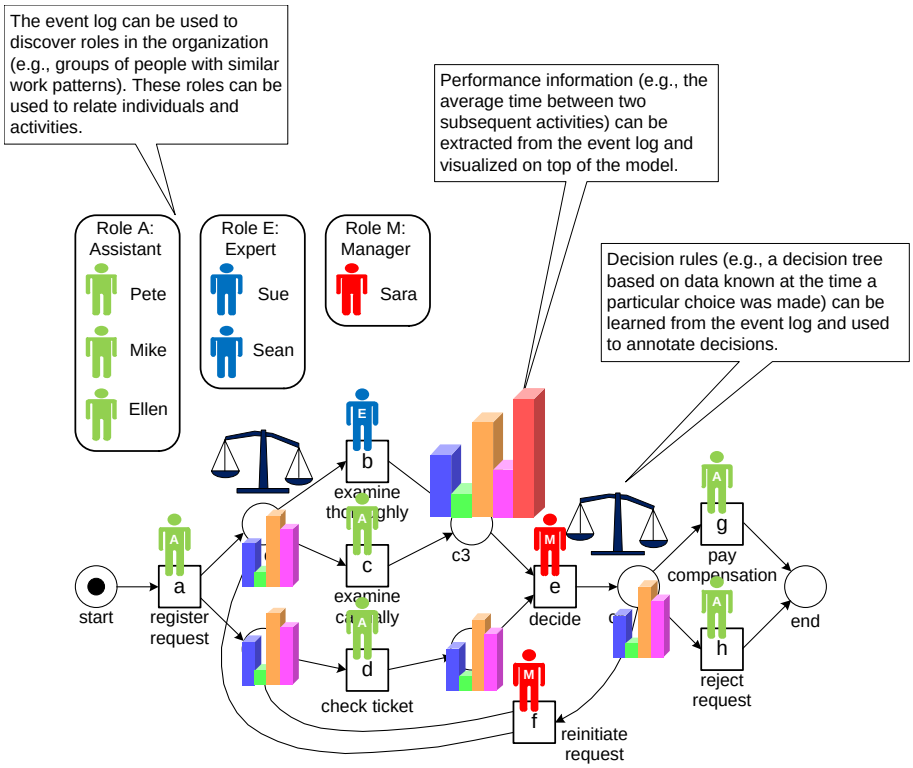


Fig. 2 The process model extended with additional perspectives: the organizational perspective ("What are the organizational roles and which resources are performing particular activities?"), the case perspective ("Which characteristics of a case influence a particular decision?"), and the time perspective ("Where are the bottlenecks in my process?")

Using a small event log, this section introduced the topic of process mining. Unlike conventional process modeling and analysis approaches, there is a direct link between model and reality recorded in the form of an event log.

The event log shown in Table 1 and the corresponding process model in Fig. 2 provide an oversimplified view of process mining. There is a continuum of processes ranging from highly structured processes (often referred to as *Lasagna* processes) to unstructured processes (*Spaghetti* processes). An example of a Spaghetti process is shown in Fig. 3. The model was obtained using the heuristic miner with default settings. Note that some low frequent behavior has already been filtered out, i.e., the real process is even more Spaghetti-like than the model shown in Fig. 3. This illustrates the challenges related to process mining when applied to less structured processes.

Process models such as the one depicted in Fig. 3 provide a "reality check" for business consultants, process analysts, and IT developers.

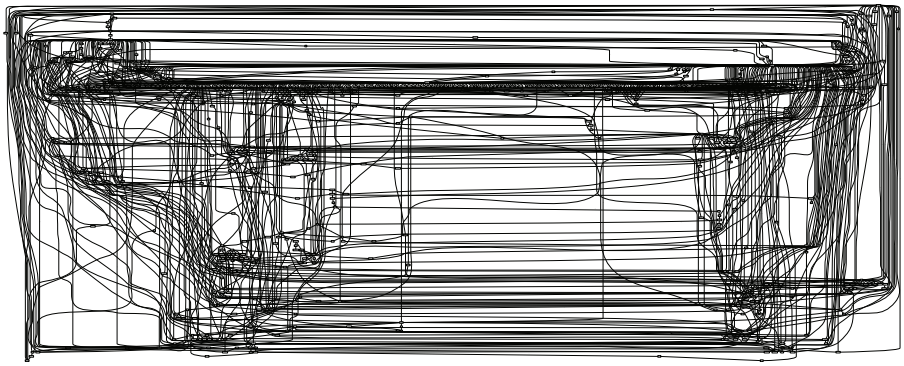


Fig. 3 Spaghetti process describing the diagnosis and treatment of 2765 patients in a Dutch hospital [3]. The process model was constructed based on an event log containing 114,592 events. There are 619 different activities (taking event types into account) executed by 266 different individuals (doctors,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

We have developed various techniques to simplify such models and extract meaningful knowledge from them. In fact, it radically changed our understanding of process modeling and process models.

3 Play-In, Play-Out, and Replay

One of the key elements of process mining is the emphasis on establishing a strong relation between a process model and "reality" captured in the form of an event log [3]. Inspired by the terminology used by David Harel in the context of Live Sequence Charts [13], we use the terms *Play-In*, *Play-Out*, and *Replay* to reflect on this relation. Figure 4 illustrates these three notions.

Play-Out refers to the classical use of process models. For example, given a Petri net or BPMN model, it is possible to generate behavior. The traces in Table 1 could have been obtained by repeatedly "playing the token game" using the Petri net of Fig. 1. Play-Out can be used both for the analysis and the enactment of business processes. A workflow engine can be seen as a "Play-Out engine" that controls cases by only allowing the "moves" possible according to the model. Hence, Play-Out can be used to enact operational processes using some executable model. Simulation tools also use a Play-Out engine to conduct experiments. The main idea of simulation is to repeatedly run a model and thus collect statistics and confidence intervals. Note that a simulation engine is similar to a workflow engine. The main difference is that the simulation engine interacts with a modeled environment whereas the workflow engine interacts with the real environment (workers, customers, etc.). Also classical verification approaches using exhaustive state-space analysis – often referred to as model checking [12] – can be seen as Play-Out methods.

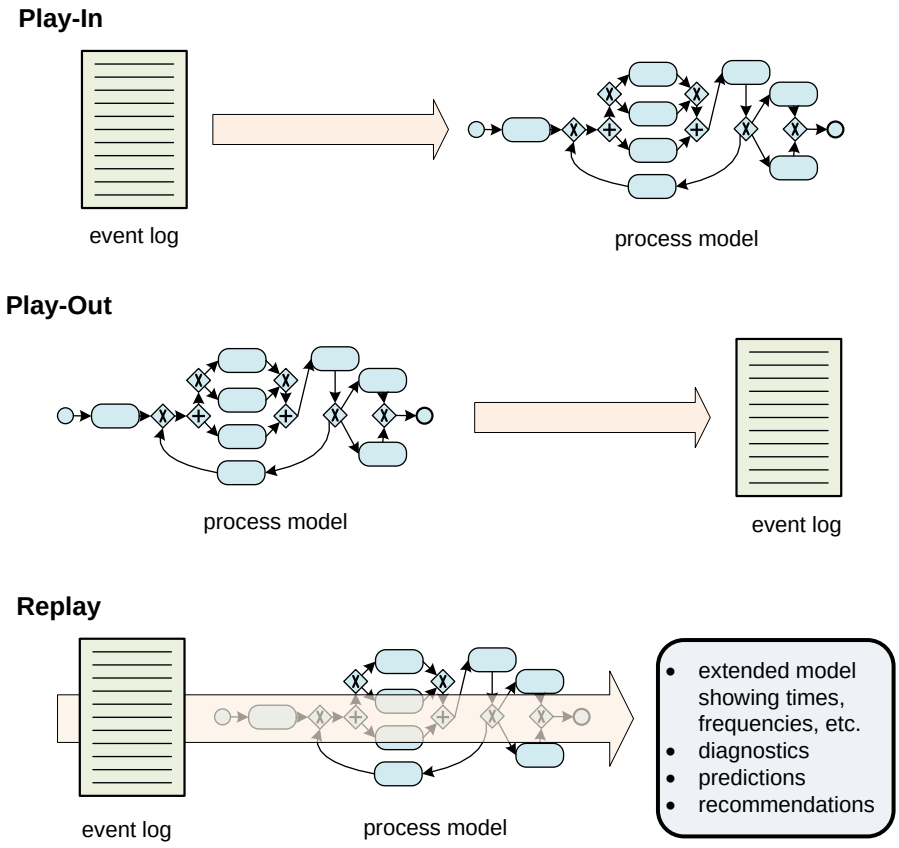


Fig. 4 Three ways of relating event logs (or other sources of information containing example behavior) and process models: *Play-In*, *Play-Out*, and *Replay*

Play-In is the opposite of Play-Out, i.e., example behavior is taken as input and the goal is to construct a model. Play-In is often referred to as *inference*. The α-algorithm and other process discovery approaches are examples of Play-In techniques. Note that the Petri net of Fig. 1 can be derived automatically given an event log like the one in Table 1. Most data mining techniques use Play-In, i.e., a model is learned on the basis of examples. However, traditionally, data mining has not been concerned with process models [3]. Typical examples of models that are considered by data mining techniques are decision trees ("people that drink more than five glasses of alcohol and smoke more than 56 cigarettes tend to die young") and association rules ("people that buy diapers also buy beer") [23]. Unfortunately, it is not possible to use conventional data mining techniques to Play-In process models. Only recently, process mining techniques have become readily available to discover process models based on event logs.

Replay uses an event log *and* a process model as input. The event log is "replayed" on top of the process model. As shown earlier, it is possible to replay trace $\langle a, b, d, e, h\rangle$ on the Petri net in Fig. 1; simply "play the token game" by forcing the transitions to fire (if possible) in the order indicated. An event log may be replayed for different purposes:

- *Conformance checking*: discrepancies between the log and the model can be detected and quantified by replaying the log. For instance, replaying trace $\langle a, b, e, h\rangle$ on the Petri net in Fig. 1 will show that d should have happened but did not (cf. alignment γ_1 with g replaced by h).
- *Extending the model with frequencies and temporal information.* By replaying the log one can see which parts of the model are visited frequently. Replay can also be used to detect bottlenecks. Consider, for example, the trace $\langle a^8, b^9, d^{20}, e^{21}, h^{21}\rangle$ in which the superscripts denote timestamps. By replaying the trace on top of Fig. 1 one can see that e was enabled at time 20 and occurred at time 21. The enabling of e was delayed by the time it took to complete d; although d was enabled already at time 8, it occurred only at time 20.
- *Constructing predictive models.* By replaying event logs one can build predictive models, i.e., for the different states of the model particular predictions can be made. For example, a predictive model learned by replaying many cases could show that the expected time until completion after enabling e is eight hours.
- *Operational support.* Replay is not limited to historic event data. One can also replay partial traces of cases still running. This can be used for detecting deviations at run-time, e.g., the partial trace $\langle a^8, e^{11}\rangle$ of a case that is still running will never fit into Fig. 1. Hence, an alert can be generated before the case completes. Similarly, it is possible to predict the remaining processing time or the likelihood of being rejected of a case having a partial trace, e.g., a partial executed case $\langle a^8, b^9\rangle$ has an expected remaining processing time of 3.5 days and a 40 percent probability of being rejected. Such predictions can also be used to recommend suitable next steps to progress the case.

Until recently, the focus of process analysis was mostly on modeled processes rather than factual data. However, practical experiences with Play-In and Replay shed new light on the quality of process models. The next section summarizes some of the lessons learned.

Clearly, process mining is only possible if it is possible to obtain enough high-quality event data. Events may be stored in database tables, message logs, mail archives, transaction logs, and other data sources. More important than the storage format, is the *quality* of such event logs. The quality of a process mining result heavily depends on the input. Therefore, event logs should be treated as *first-class citizens* in the information systems supporting the processes to be analyzed. Unfortunately, event logs are often merely a "by-product" used for debugging or profiling. Moreover, as event logs contain only sample behavior, they should not be assumed to be com-

plete. Process mining techniques need to deal with incompleteness by using an "open world assumption": the fact that something did not happen does not imply that it cannot happen. This makes it challenging to deal with small event logs with a lot of variability. Despite these challenges it is obvious that there is an exponential growth of event data and that the quality of event data continue to improve as IT systems become more pervasive.

4 Seven Problems Related to Process Modeling

As indicated earlier, we have applied process mining (in particular our process mining tool ProM) in more than 100 organizations [3]. This changed the way we view models and provides useful input for answering the question *"What Makes a Good Process Model?"*. In particular, we noted typical limitations of process models and errors frequently made when modeling a process. In the remainder, we identify some root causes for the ineffectiveness or futility of many business process models based on insights obtained through process mining.

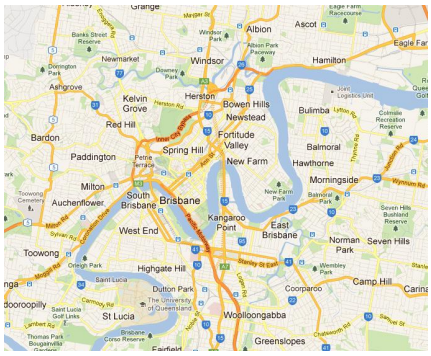
4.1 First Problem: Aiming for one model that suits all purposes

Often modelers aim to create one "perfect process model" that needs to serve all possible purposes. However, a process model is merely a view on the process. Depending on the questions that need to be answered, different views may be needed. Nevertheless, designers often look for a kind of "holy grail", i.e., the process model that can be used to answer all questions. Obviously, such a model does not exist.

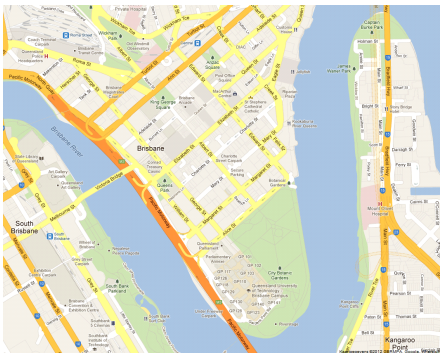
When discovering process models on the basis of event logs, it becomes evident that process models should be seen as *maps* (cf. Figure 5). There may be highway maps, city maps, bicycle maps, boating maps, and hiking maps covering (parts of) the same area. Some elements may not be shown while other elements are emphasized. Some maps may show a larger area with less detail (cf. Figure 5(a)) whereas other maps show a smaller area with more details (cf. Figure 5(b)). When cycling another map is desired showing the bicycle paths (cf. Figure 5(c)). Similarly, depending on the intended purpose (discussion, bottleneck analysis, auditing, simulation, etc.), different process models are needed. Therefore, one should not focus on the quality of single maps but concentrate on the ability to generate different maps that suit specific purposes.

4.2 Second Problem: Straitjacketing smaller interacting processes into one monolithic model

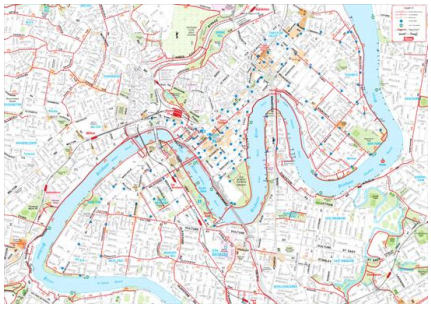
Existing business process modeling notations such as BPMN, BPEL, UML activity diagrams, YAWL and WF-nets, are mostly used to describe the



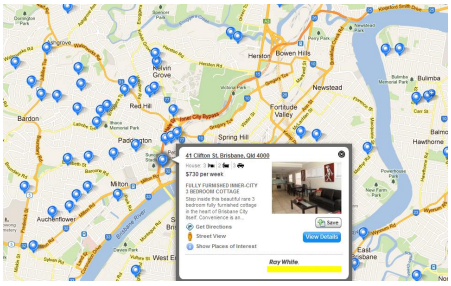
(a) map of Brisbane (zoomed-out)



(b) map of Brisbane (zoomed-in)



(c) bicycle map showing bicycle paths



(d) map showing real-estate for sale

(e) map showing traffic jams (red roads are congested, green roads show flowing traffic)

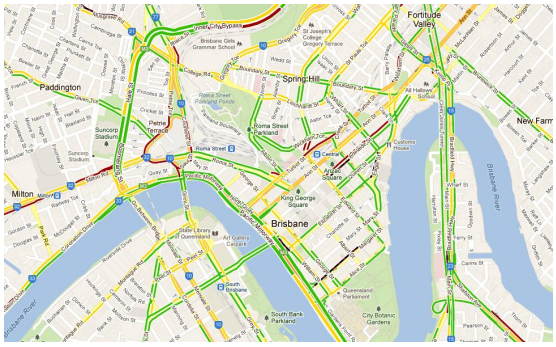


Fig. 5 Process models can be viewed as maps. For the same process there may be multiple maps (depending on the desired level of detail and use). Moreover, information may be projected on such maps thus "breathing life" into otherwise static/abstract views on reality

life-cycle of one process instance in isolation. We refer to such models as *monolithic*. Note that a monolithic model may be hierarchical, i.e., activities at one level are decomposed into subprocesses. However, all activities at all the different levels refer to status changes of the same process instance. Of course a process may be instantiated multiple times, but the interaction between instances (other than the competition for resources) is not modeled.

Monolithic process models are typically unable to adequately capture real-life processes

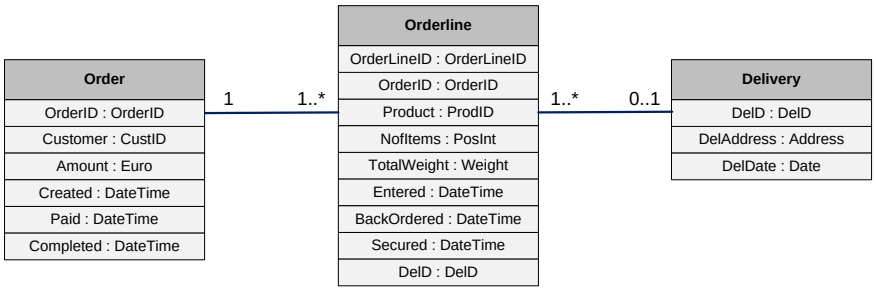


Fig. 6 Class diagram showing the relations between orders, order lines, and deliveries

To illustrate the problem consider the class diagram in shown in Fig. 6. Orders consist of one or more orderlines. An order may be decomposed into multiple deliveries and a delivery may be composed of orderlines originating from different orders of the same customer. For example, a customer places an order for three books. One week later, she places an order for two more books. Of the books ordered initially, only two can be delivered immediately. The book that could not be delivered in the first week is delivered together with the two books ordered in the second week. This shows that there is a many-to-many relationship between orders and deliveries. There are different types of events not shown in Fig. 6. For example, "customer places an order", "customer pays for the order", "availability of orderline is checked", "delivery is shipped", etc. It is relatively easy to model the life-cycles of orders, orderlines, and deliveries in separate diagrams. However, mainstream notations do not allow for the modeling of the interactions between these individual life-cycles. For example, in BPMN one needs to straitjacket things into one diagram modeling a single instance in isolation. Of course a BPMN diagram describes a process at the type level, i.e., the process may be instantiated for many instances. However, interactions between instances are not modeled. (Note that we are not referring to message flows connecting different pools in BPMN [18].) Modern workflow languages like YAWL support so-called "multiple instance tasks". This allows for the embedding of "smaller instances" (e.g., orderlines) into "larger instances" (e.g., orders). However, this does not solve the problem of faithfully modeling the many-to-many relationship between orders and deliveries. The first workflow language not forcing the designer to straitjacket smaller interacting processes into one monolithic model was the *Proclets* language [5]. Recently, such notations gained more attention and are now commonly referred to as "artifact centric" [9].

The problem just described surfaces when applying process mining techniques to end-to-end processes using data from systems such as SAP, Oracle,

and Peoplesoft [3]. Nevertheless, process modelers try to squeeze such processes into a monolithic model. A nice example can be found in SAP's Reference Model which describes more than 600 standard processes supported by the SAP system [16]. One of these processes deals with the handling of job applications and the management of open positions. In SAP's process model some of the activities deal with a single job application whereas others deal with a single open position. This is put together in a single model where both types of activities are connected, i.e., job and position instances are mixed up. Obviously, this does not make any sense, e.g., there may be many applications for the same position that all need to be dealt with concurrently. This is a common mistake; good models do not mix-up different types of instances.

4.3 Third Problem: Using static hierarchical decomposition as the only abstraction mechanism

Most process modeling notations support some notion of hierarchy. In principle, such hierarchy concepts are useful. The designer can hide details at lower levels. However, often the hierarchy is static and static decomposition is the only way of abstraction. The distribution of activities over the various levels in the hierarchy cannot be varied and low-level connections surface at higher levels.

We explain this problem using the metamorphic relation between processes and maps mentioned earlier. Consider for example the zooming functionality of Google Maps. Note that, while zooming out, insignificant things are either left out or dynamically clustered into aggregate shapes (e.g., streets and suburbs amalgamate into cities). See Figure 5(a)-(b) as an illustration. Note that hierarchical decomposition cannot be used to relate both maps. Important roads (e.g., highways) may be visible at all levels whereas insignificant roads disappear or amalgamate into suburbs.

Navigation systems and applications such as Google Maps provide such a seamless zoom. Traditionally, process models are static, e.g., it is impossible to seamlessly zoom in to see part of the process in more detail. To deal with larger processes, typically a static hierarchical decomposition is used. In such a hierarchy, a process is composed of subprocesses, and in turn these subprocesses may be composed of smaller subprocesses. However, *despite the hierarchy the "true level of detail" is constant for all levels.* Activities in two different subprocesses can only be connected if there is a connection at the higher level. Therefore, one cannot abstract from details at higher levels, i.e., details are encapsulated but not removed. From a system's design perspective this can be justified easily. However, when the goal is to make understandable models, a single static hierarchical decomposition is insufficient. The hierarchy should be dynamic and the model presented should, if desired, also leave out things less relevant. Note that a geographic map may leave out many details that become only visible when zooming in.

The fact that there is no highway connecting two cities does not imply that there cannot be a dirt road connecting these cities. Yet, static hierarchical decomposition forces the designer to implicitly show the dirt road at all levels. Metaphorically speaking: the dirt road needs to be hidden under a "virtual highway" at the higher level.

Note that this problem is related to the first problem identified (Section 4.1). Classically, a model supports just one abstraction level. We advocate support for varying abstraction levels (seamless zoom and selection capabilities).

4.4 Fourth Problem: Modeling humans as if they are machines doing a single task

Process modeling is not limited to control-flow, e.g., when making a simulation model it is vital to also model resources. However, the resource perspective is often modeled in a rather shallow manner, e.g., tasks and resource are just connected through roles. Although simple mathematical models may suffice to model machines or people working in an assembly line, they are inadequate when modeling people involved in multiple processes and exposed to multiple priorities [2,6]. A worker who is involved in multiple processes needs to distribute his attention over these. This makes it difficult to model one process in isolation. Workers also do not work at constant speed. Figure 7 visualizes the so-called "Yerkes-Dodson law of arousal" that describes the relation between workload and performance of people [2,24]. In most processes one can easily observe that people typically take more time to complete a task and effectively work fewer hours per day if there is hardly any work to do. Unfortunately, such phenomena are seldom modeled thus making process models incomplete. For example, most simulation models involving human actors distributing their attention over multiple processes are unable to correctly predict response times and flow times.

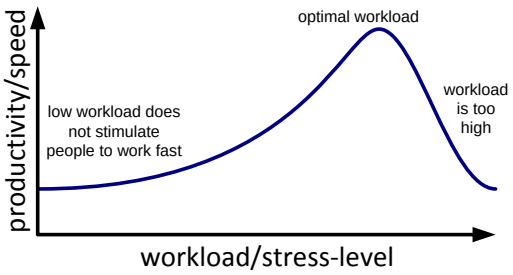


Fig. 7 Yerkes-Dodson law of arousal

Through process mining one can easily observe that people behave different from machines. For example, it may be that long waiting times are caused by workers spending just a few hours per month on the process (e.g., a manager that needs to approve each request formally). Moreover, Yerkes-Dodson law of arousal can be observed in many processes where both the start and completion times of activities are recorded.

4.5 Fifth Problem: Being vague about vagueness

In large organizations, process models often serve as the interface between "IT" and "Business". IT developers and system analysts prefer detailed and relatively precise models without any consideration for business aspects whereas domain experts and managers prefer informal models focusing on non-technical business-related aspects. As a result, the audience for a particular process model can be rather diverse. This creates all kinds of interpretation problems where one group associates more meaning to parts of the model than others. The solution is not to remove all ambiguity by enforcing the use of formal specifications; there is a clear need for *vagueness* to make models comprehensible [14]. Unfortunately, most process models do not show whether parts of it are vague or not.

Consider for example BPMN and the empirical study presented in [17]. The BPMN language offers many constructs, but most users understand and use only a small subset. An expert may read a BPMN model as if it was an executable workflow model, whereas a domain expert may view the model as just a PowerPoint diagram without any formal meaning. In other words: most models are "vague about vagueness".

Process mining can be used to get grip on the explicit representation of vagueness. We often aim at creating a model that explains 100% of the observed behavior. However, one can also aim at creating a so-called "80/20 model", i.e., the process model describes 80% of the behavior seen in the log. This model is typically relatively simple because often the remaining 20% of the log account for 80% of the variability in the process. Consider for example the process model shown Fig. 3. This model can be simplified dramatically by focusing on the most frequent activities and/or by leaving out less frequent paths.

Labeling a process model as an "80/20 model", "90/20 model" or "70/25 model" makes vagueness explicit and allows people to focus on the mainstream behavior. Moreover, using conformance checking one can indicate where the simplified model deviates from reality, i.e., vagueness can be quantified and located.

4.6 Sixth Problem: Abstracting from the things that really matter

Process model notations tend to focus on the "process logic", i.e., the ordering of activities. Some notations also allow for the modeling of other

perspectives such as the resource perspective (e.g., showing roles) and the data perspective (e.g., conditions on arcs). However, these are all design-oriented features and not necessarily the things that matter most to the person looking at the model. Costs, frequencies, utilization, and other performance related metrics are of the utmost importance for process models but rarely shown. Some authors distinguish between functional (process logic, work distribution, etc.) and non-functional (costs, response times, frequencies, quality) elements. The focus of most process models is on functional elements thereby neglecting non-functional aspects. In fact, we argue that lion's share of models abstract from the aspects that matter most.

To illustrate this viewpoint, let us consider the notion of *concept drift* identified in the context of process mining [11]. The term concept drift refers to the situation in which the process is changing while being analyzed. For instance, in the beginning of the event log two activities may be concurrent whereas later in the log these activities become sequential. There may also be seasonal influences causing long waiting times in particular periods. The arrival process of new process instances or the availability of resources may fluctuate. Such phenomena are seldom shown in process models. Yet, these "second-order dynamics" are important for most stakeholders.

Process models should show what matters most. However, when looking at notations such as BPMN, the focus seems to be on things that are less relevant.

4.7 Seventh Problem: Color, size and location without meaning

As indicated before, process models can be viewed as maps. In Section 4.3, we showed that, by zooming out, less relevant details are abstracted from. However, cartographers not only eliminate irrelevant details, but also use colors to highlight important features. For instance, a map may emphasize the importance of highways using the color red. Moreover, graphical elements have a particular size to indicate their significance, e.g., the sizes of lines and dots may vary. For instance, the size of a city name is proportional to the number of citizens. Geographical maps also have a clear interpretation of the x-axis and y-axis, i.e., the layout of a map is not arbitrary as the coordinates of elements have a meaning. *All of this is in stark contrast with mainstream process models.* The x-axis and y-axis of a process model have no meaning, e.g., the layout does not add any information. Although modeling tools allow for using colors, the color typically has no semantics. The different types of model elements (e.g., activities, gateways, events, connectors, and places) typically have a default color. Moreover, the size of a model element also has no semantics. Typically all elements of a particular type have the same size. Because size, color, and layout are not employed when creating process maps, the result is less intuitive and less informative. However, ideas from cartography can easily be incorporated in the construction of business process maps. Some examples:

- The *size* of an activity can reflect its frequency or some other property indicating its significance (e.g., costs or resource use).
- The *color* of an activity can reflect the mean service time of the activity. For example, activities that take longer than average are colored red whereas short running activities are colored green.
- The *width* of an arc can reflect the importance of the corresponding causal dependency.
- The *coloring* of arcs can be used to highlight bottlenecks.
- The *positioning* of activities can have a well-defined meaning. Similar to swimlanes the y-axis could reflect the role associated to an activity. Comparable to a Gantt chart, the x-axis could reflect some temporal aspect.

In Section 4.6 we argued that process models often abstract from the things that really matter. The above examples show that models can express important non-functional elements graphically. However, it is important to use these conventions in a consistent manner across different maps.

Note that in addition to showing important non-functional elements graphically, the understandability of process models is an important issue often limiting the use of complex process models. In [19, 20] various patterns are presented that can be used to reduce the complexity of process models by modifying the abstract and concrete syntax of a process model.

5 Breathing Life Into Process Models

In this paper, we posed the question "What Makes a Good Process Model?". To address the question we did not propose a new notation or methodology. Instead we exposed problems related to the way process models are constructed and used. These problems were identified while conducting dozens of process mining projects. Process mining sheds a new light on the role of models. The automatic generation of process models (Play-In) shows that one should *not* aim at the creation of a single model; instead one should focus on the *capability to generate appropriate models based on the questions at hand*. The direct connection between event log and process model (Replay) can be exploited to show deviations and to enrich the models with information that really matters (e.g., showing bottlenecks).

Experiences with process mining suggest that process models should be seen as the "maps" describing the operational processes of organizations. The "map metaphor" was used to illustrate various problems. Similarly, information systems can be looked at as "navigation systems" guiding the flow of work in organizations. Based on maps of the organization and its processes, the information systems should assist its users in reaching desired "destinations".

Unfortunately, many organizations fail in creating and maintaining accurate process maps. Often process models are outdated and have little to do with reality. Moreover, most information systems fail to provide the

functionality offered by todays navigation systems. For instance, workers are not guided by the information system and need to work behind the system's back to get things done. In fact, useful information such as the "estimated arrival time" of a running case is not provided. However, process mining techniques providing operational support (e.g., predictions and recommendations) [1,3] can be used to realize such functionality.

Process mining can be used to breathe life into process models. See for example Figure 5(d) showing a map with real estate for sale and Figure 5(e) showing Brisbane's traffic jams. Similarly, event data can be used to show congestion in business processes.

By establishing a close connection between process maps and the actual behavior recorded in event logs, it is possible to realize information system with TomTom-like functionality [1]. Analogous to TomTom's navigation devices, process mining tools can help end users (a) by navigating through processes (e.g., zoom in and zoom out), (b) by projecting dynamic information on process maps (e.g., showing "traffic jams" in business processes), and (c) by providing predictions regarding running cases (e.g., estimating the "arrival time" of a case that is delayed). We refer the reader to the process mining tool ProM that operationalizes these ideas [3].

Acknowledgements The author thanks all that contributed to development of ProM. The application of process mining to numerous real-life processes led to insights reported in this paper.

References

1. W.M.P. van der Aalst. TomTom for Business Process Management (TomTom4BPM). In P. van Eck, J. Gordijn, and R. Wieringa, editors, *Advanced Information Systems Engineering, Proceedings of the 21st International Conference on Advanced Information Systems Engineering (CAiSE'09)*, volume 5565 of *Lecture Notes in Computer Science*, pages 2–5. Springer-Verlag, Berlin, 2009.
2. W.M.P. van der Aalst. Business Process Simulation Revisited. In J. Barjis, editor, *Enterprise and Organizational Modeling and Simulation*, volume 63 of *Lecture Notes in Business Information Processing*, pages 1–14. Springer-Verlag, Berlin, 2010.
3. W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, Berlin, 2011.
4. W.M.P. van der Aalst, A. Adriansyah, and B. van Dongen. Replaying History on Process Models for Conformance Checking and Performance Analysis. *WIREs Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
5. W.M.P. van der Aalst, P. Barthelmess, C.A. Ellis, and J. Wainer. Proclets: A Framework for Lightweight Interacting Workflow Processes. *International Journal of Cooperative Information Systems*, 10(4):443–482, 2001.
6. W.M.P. van der Aalst, J. Nakatumba, A. Rozinat, and N. Russell. Business Process Simulation. In J. vom Brocke and M. Rosemann, editors, *Handbook on Business Process Management*, International Handbooks on Information Systems, pages 313–338. Springer-Verlag, Berlin, 2010.

7. W.M.P. van der Aalst, H.A. Reijers, and M. Song. Discovering Social Networks from Event Logs. *Computer Supported Cooperative work*, 14(6):549–593, 2005.
8. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
9. ACSI. Artifact-Centric Service Interoperation (ACSI) Project Home Page. `www.acsi-project.eu`.
10. A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guzar, N. Kartha, C.K. Liu, R. Khalaf, Dieter Koenig, M. Marin, V. Mehta, S. Thatte, D. Rijn, P. Yendluri, and A. Yiu. Web Services Business Process Execution Language Version 2.0 (OASIS Standard). WS-BPEL TC OASIS, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, 2007.
11. R.P. Jagadeesh Chandra Bose, W.M.P. van der Aalst, I. Zliobaite, and M. Pechenizkiy. Handling Concept Drift in Process Mining. In H. Mouratidis and C. Rolland, editors, *International Conference on Advanced Information Systems Engineering (Caise 2011)*, volume 6741 of *Lecture Notes in Computer Science*, pages 391–405. Springer-Verlag, Berlin, 2011.
12. E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts and London, UK, 1999.
13. D. Harel and R. Marelly. *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer-Verlag, Berlin, 2003.
14. T. Herrmann, M. Hoffmann, K.U. Loser, and K. Moysich. Semistructured models are surprisingly useful for user-centered design. In G. De Michelis, A. Giboin, L. Karsenty, and R. Dieng, editors, *Designing Cooperative Systems (Coop 2000)*, pages 159–174. IOS Press, Amsterdam, 2000.
15. IEEE Task Force on Process Mining. Process Mining Manifesto. In *BPM Workshops*, volume 99 of *Lecture Notes in Business Information Processing*. Springer-Verlag, Berlin, 2011.
16. J. Mendling, H.M.W. Verbeek, B.F. van Dongen, W.M.P. van der Aalst, and G. Neumann. Detection and Prediction of Errors in EPCs of the SAP Reference Model. *Data and Knowledge Engineering*, 64(1):312–329, 2008.
17. M. Zur Muehlen and J. Recker. How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In Z. Bellahsene and M. Léonard, editors, *Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'08)*, volume 5074 of *Lecture Notes in Computer Science*, pages 465–479. Springer-Verlag, Berlin, 2008.
18. OMG. Business Process Model and Notation (BPMN). Object Management Group, formal/2011-01-03, 2011.
19. M. La Rosa, A.H.M. ter Hofstede, P. Wohed, H.A. Reijers, J. Mendling, and W.M.P. van der Aalst. Managing Process Model Complexity via Concrete Syntax Modifications. *IEEE Transactions on Industrial Informatics*, 7(2):255–265, 2011.
20. M. La Rosa, P. Wohed, J. Mendling, A.H.M. ter Hofstede, H.A. Reijers, and W.M.P. van der Aalst. Managing Process Model Complexity via Abstract Syntax Modifications. *IEEE Transactions on Industrial Informatics*, 7(4):614–629, 2011.
21. A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, 2008.

22. M. Song and W.M.P. van der Aalst. Towards Comprehensive Support for Organizational Mining. *Decision Support Systems*, 46(1):300–317, 2008.
23. I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann, 2005.
24. R.M. Yerkes and J.D. Dodson. The Relation of Strength of Stimulus to Rapidity of Habit-Formation. *Journal of Comparative Neurology and Psychology*, 18:459–482, 1908.