Conformance testing : measuring the fit and appropriateness of event logs and process models

Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models

A. Rozinat[1,2] and W.M.P. van der Aalst[1]

[1] Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
w.m.p.v.d.aalst@tm.tue.nl
[2] Department of Business Process Technology, HPI University of Potsdam
P.O. Box 900460, D-14440, Potsdam, Germany.
a.rozinat@tm.tue.nl

Abstract Most information systems log events (e.g., transaction logs, audit trails) to audit and monitor the processes they support. At the same time, many of these processes have been explicitly modeled. For example, SAP R/3 logs events in transaction logs and there are EPCs (Event-driven Process Chains) describing the so-called reference models. These reference models describe how the system should be used. The co-existence of event logs and process models raises an interesting question: "Does the event log *conform* to the process model and vice versa?". This paper demonstrates that there is not a simple answer to this question. To tackle the problem, we distinguish two dimensions of conformance: *fitness* (the event log may be the result of the process modeled) and *appropriateness* (the model is a likely candidate from a structural and behavioral point of view). Different metrics have been defined and a *Conformance Checker* has been implemented within the ProM Framework.

1 Introduction

New legislation such as the Sarbanes-Oxley (SOX) Act [15] and increased emphasis on corporate governance and operational efficiency has triggered the need for improved auditing systems. To audit an organization, business activities need to be monitored. Buzzwords such as BAM (Business Activity Monitoring), BOM (Business Operations Management), BPI (Business Process Intelligence) illustrate the interest of vendors to support the monitoring and analysis of business activities. The close monitoring of processes can be seen as a second wave following the wave of business process modeling and simulation. In the first wave the emphasis was on constructing process models and analyzing them. The many notations (e.g., Petri nets, UML activity diagrams, EPCs, IDEF, BPMN, and not to mention the vendor or system specific notations) illustrate this. This creates the interesting situation where processes are being monitored while at the same time there are process models describing these processes. The focus of this paper is on *conformance*, i.e., "Is there a good match between the recorded events and the model?". A term that could be used in this context is "business alignment", i.e., are the real process (reflected by the log) and the process model (e.g., used to configure the system) aligned properly.

Most information systems, such as WFM, ERP, CRM, SCM, and B2B systems, provide some kind of *event log* (also referred to as transaction log or audit trail) [5]. Typically such an event log registers the start and/or completion of activities. Every event refers to a case (i.e., process instance) and an activity, and, in most systems, also a timestamp, a performer, and some additional data. In this paper, we only use the first two attributes of an event, i.e., the identity of the case and the name of the activity. Meanwhile, any organization documents its processes in some form. The reasons for making these process models are manifold. Process models are used for communication, ISO 9000 certification, system configuration, analysis, simulation, etc. A process model may be of a *descriptive* or of a *prescriptive* nature. Descriptive models try to capture existing processes without being normative. Prescriptive models describe the way that processes should be executed. In a Workflow Management (WFM) system prescriptive models are used to enforce a particular way of working using IT [2]. However, in most situations prescriptive models are not used directly by the information system. For example, the reference models in the context of SAP R/3 [12] and ARIS [16] describe the "preferred" way processes should be executed. People actually using SAP R/3 may deviate from these reference models.

In this paper, we will use Petri nets [9] to model processes. Although the metrics are based on the Petri net approach, the results of this paper in general can be applied to any modeling language that can be equipped with executable semantics. An event log is represented by a set of event sequences, also referred to as traces. Each case in the log refers to one sequence. The most dominant requirement for conformance is *fitness*. An event log and Petri net "fit" if the Petri net can generate each trace in the log. In other words: the Petri net should be able to "parse" every event sequence. We will show that it is possible to quantify fitness, e.g., an event log and Petri net may have a fitness of 0.66. Unfortunately, a good fitness does not imply conformance. As we will show, it is easy to construct Petri nets that are able to parse any event log. Although such Petri nets have a fitness of 1 they do not provide meaningful information. Therefore, we introduce a second dimension: *appropriateness*. Appropriateness tries to capture the idea of *Occam's razor*, i.e., "one should not increase, beyond what is necessary, the number of entities required to explain anything". Clearly, this dimension is not as easy to quantify as fitness. We will distinguish between *structural appropriateness* (if a simple model can explain the log, why choose a complicated one) and *behavioral appropriateness* (the model should not be too generic). Using examples, we will show that both the structural and behavioral aspects need to be considered to measure appropriateness adequately.

To actually measure conformance, we have developed a tool called *Conformance Checker*. It is part of the *ProM framework*[3], which offers a wide range of tools related to process mining, i.e., extracting information from event logs [5].

[3] Both documentation and software can be downloaded from *www.processmining.org*.

This paper is organized as follows. Section 2 introduces a running example that will be used to illustrate the concept of conformance. Section 3 discusses the need for two dimensions. The fitness dimension is discussed in Section 4. The appropriateness dimension is elaborated in Section 5. Section 6 shows how these properties can be verified using the conformance checker in ProM. Finally, some related work is discussed and the paper is concluded.

2 Running Example

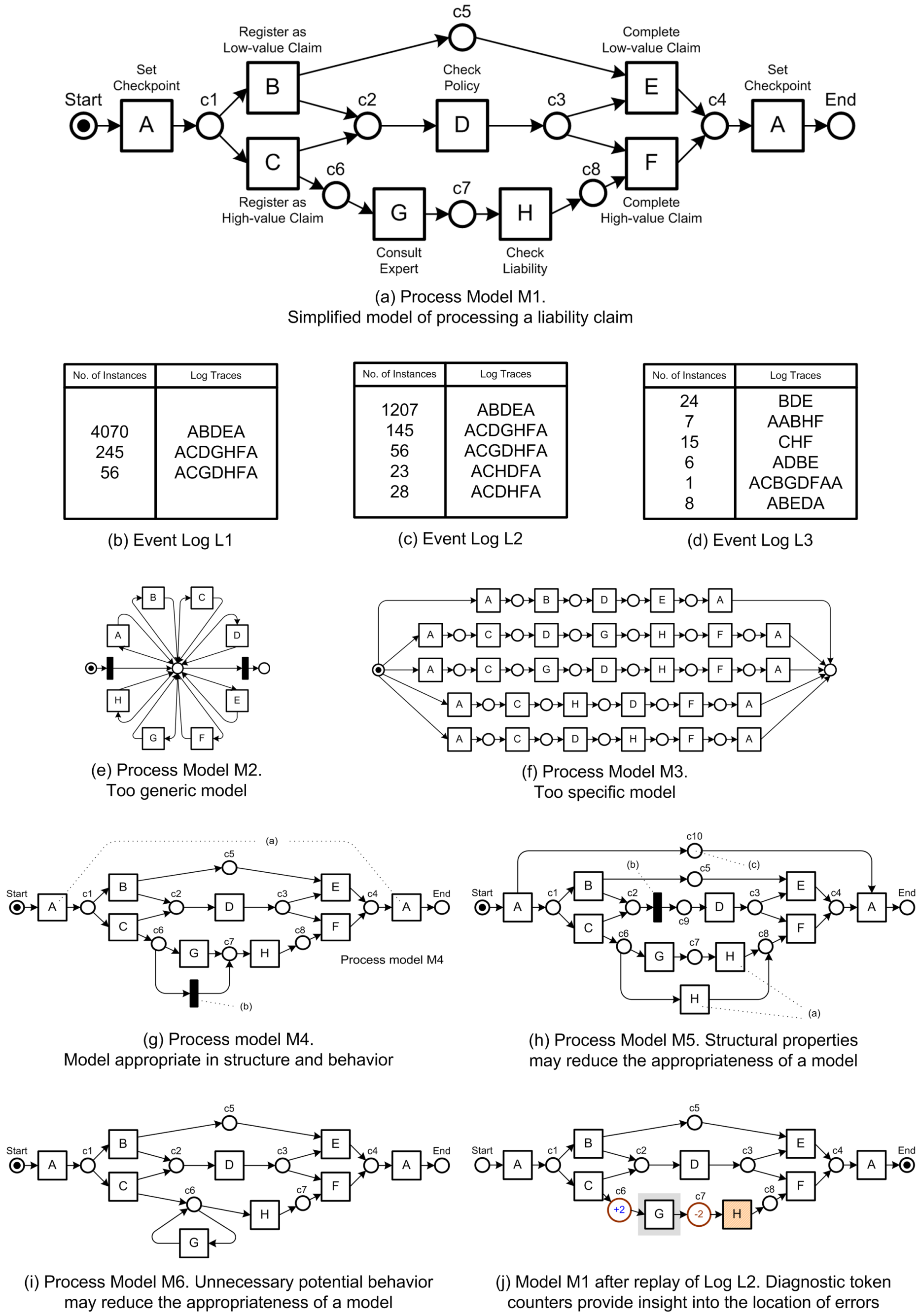
The example model used throughout the paper concerns the processing of a liability claim within an insurance company (cf. Figure 1(a)).

At first there are two tasks bearing the same label "Set Checkpoint". This can be thought of as an automatic backup action within the context of a transactional system, i.e., activity A is carried out at the beginning to define a rollback point enabling atomicity of the whole process, and at the end to ensure durability of the results. Then the actual business process is started with the distinction of low-value claims and high-value claims, which get registered differently (B or C). The policy of the client is checked anyway (D) but in the case of a high-value claim, additionally, the consultation of an expert takes place (G), and then the filed liability claim is being checked in more detail (H). Finally, the claim is completed according to the former choice between B and C (i.e., E or F).

Figures 1(b)-(d) show three example logs for the process described in Figure 1(a) at an aggregate level. This means that process instances exhibiting the same event sequence are combined as a logical log trace, memorizing the number of instances to weigh the importance of that trace. That is possible since only the control flow perspective is considered here. In a different setting like, e.g., mining social networks [4], the resources performing an activity would distinguish those instances from each other.

3 Two Dimensions of Conformance: Fitness and Appropriateness

Measurement can be defined as a set of rules to assign values to a real-world property, i.e., observations are mapped onto a numerical scale. In the context of conformance testing this means to weigh the "distance" between the behavior described by the process model and the behavior actually observed in the workflow log. If the distance is zero, i.e., the real business process exactly matches the specified behavior, one can say that the log *fits* the model. With respect to the example model *M1* this applies for event log *L1*, since every log trace can be associated with a valid path from *Start* to *End*. In contrast, event log *L2* does not match completely as the traces $ACHDFA$ and $ACDHFA$ lack the execution of activity G, while event log *L3* does not even contain one trace corresponding to the specified behavior. Somehow *L3* seems to fit "worse" than *L2*, and the degree of fitness should be determined according to this intuitive notion of conformance, which might vary for different settings.



(a) Process Model M1.
Simplified model of processing a liability claim

No. of Instances	Log Traces
4070	ABDEA
245	ACDGHFA
56	ACGDHFA

(b) Event Log L1

No. of Instances	Log Traces
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	ACHDFA
28	ACDHFA

(c) Event Log L2

No. of Instances	Log Traces
24	BDE
7	AABHF
15	CHF
6	ADBE
1	ACBGDFAA
8	ABEDA

(d) Event Log L3

(e) Process Model M2.
Too generic model

(f) Process Model M3.
Too specific model

(g) Process model M4.
Model appropriate in structure and behavior

(h) Process Model M5. Structural properties may reduce the appropriateness of a model

(i) Process Model M6. Unnecessary potential behavior may reduce the appropriateness of a model

(j) Model M1 after replay of Log L2. Diagnostic token counters provide insight into the location of errors

Figure 1. Example models and logs

But there is another interesting — rather qualitative — dimension of conformance, which can be illustrated by relating the process models *M2* and *M3*, shown in Figure 1(e) and (f), to event log *L2*. Although the log fits both models quantitatively, i.e., the event streams of the log and the model can be matched perfectly, they do not seem to be *appropriate* in describing the insurance claim administration.

The first one is much too generic as it covers a lot of extra behavior, allowing for arbitrary sequences containing the activities A, B, C, D, E, F, G, or H, while the latter does not allow for more sequences than those having been observed but only lists the possible behavior instead of expressing it in a meaningful way. Therefore, it does not offer a better understanding than can be obtained by just looking at the aggregated log. We claim that a "good" process model should somehow be minimal in structure to clearly reflect the described behavior, in the following referred to as *structural appropriateness*, and minimal in behavior to represent as closely as possible what actually takes place, which will be called *behavioral appropriateness*.

Apparently, conformance testing demands for two different types of metrics, which are:

- *Fitness*, i.e., the extent to which the log traces can be associated with execution paths specified by the process model, and
- *Appropriateness*, i.e., the degree of accuracy in which the process model describes the observed behavior, combined with the degree of clarity in which it is represented.

4 Measuring Fitness

Different ways are conceivable to measure the fit between event logs and process models. A rather naive approach would be to generate all execution sequences allowed by the model and then compare them to the log traces using string distance metrics. Unfortunately the number of firing sequences increases very fast if a model contains parallelism and might even be infinite if we allow for loops. Therefore, this is of limited applicability.

Another possibility is to replay the log in the model and somehow measure the mismatch, which subsequently is described in more detail. The replay of every logical log trace starts with marking the initial place in the model and then the transitions that belong to the logged events in the trace are fired one after another. While doing so we count the number of tokens that had to be created artificially (i.e., the transition belonging to the logged event was not enabled and therefore could not be *successfully executed*) and the number of tokens that had been left in the model, which indicates the process not having *properly completed*.

Let k be the number of *different* traces from the aggregated log, n the number of process instances combined as one of these traces, m the number of missing tokens, r the number of remaining tokens, c the number of consumed tokens, and p the number of produced tokens during log replay, then the token-based fitness metrics f is formalized as follows.

$$f = \frac{1}{2}(1 - \frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i c_i}) + \frac{1}{2}(1 - \frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} n_i p_i}) \tag{1}$$

Note that, for all i, $m_i \leq c_i$ and $r_i \leq p_i$, and therefore $0 \leq f \leq 1$. Using the metrics f we can now calculate the fitness between the event logs *L1*, *L2*, *L3*, and the process description *M1*, respectively. The first event log *L1* shows three different log traces that all correspond to possible firing sequences of the Petri net with one initial token in the *Start* place. Thus, there are neither tokens left nor missing in the model during log replay and the fitness measurement yields $f(M1, L1) = 1$. Replaying the event log *L2* fails for the last two traces *ACHDFA* and *ACDHFA*, since the model requires activity *G* being performed before activating task *H*. Therefore, in both cases one token remains in place *c6*, and one token needs to be created artificially in place *c7* for firing transition *H* (i.e., $m_1 = r_1 = m_2 = r_2 = m_3 = r_3 = 0$, and $m_4 = r_4 = m_5 = r_5 = 1$). Counting the tokens being produced and consumed in the Petri net model (i.e., $c_1 = p_1 = 7$, and $c_2 = c_3 = p_2 = p_3 = 9$, and $c_4 = c_5 = p_4 = p_5 = 8$), and with the number of process instances per trace, given in Figure 1(c), the fitness can be measured as $f(M1, L2) \approx 0.995$. For the last event log *L3* the fitness measurement yields $f(M1, L3) \approx 0.540$.

Besides measuring the degree of fitness pinpointing the site of mismatch is crucial for giving useful feedback to the analyst. In fact, the place of missing and remaining tokens during log replay can provide insight into problems, such as Figure 1(j) visualizes some diagnostic information obtained for event log *L2*. Because of the remaining tokens (whose amount is indicated by a + sign) in place *c6* transition *G* has stayed enabled, and as there were tokens missing (indicated by a − sign) in place *c7* transition *H* has failed seamless execution.

Note that this replay is carried out in a non-blocking way and from a log-based perspective, i.e., for each log event in the trace the corresponding transition is fired, regardless whether the path of the model is followed or not. This leads to the fact that — in contrast to directly comparing the event streams of models and logs — a concatenation of missing log events is punished by the fitness metrics f just as much as a single one, since it could always be interpreted as a missing link in the model.

As a prerequisite of conformance analysis model tasks must be associated with the logged events, which may result in *duplicate tasks*, i.e., multiple tasks that are mapped onto the same kind of log event, and *invisible tasks*, i.e., tasks that have no corresponding log event. Duplicate tasks cause no problems during log replay as long as they are not enabled at the same time and can be seamlessly executed, but otherwise one must enable and/or fire the right task for progressing properly. Invisible tasks are considered to be lazy, i.e., they are only fired if they can enable the transition in question. In both cases it is necessary to partially explore the state space of the model but a detailed description is beyond the scope of this paper.

5 Measuring Appropriateness

Generally spoken, determining the degree of appropriateness of a workflow process model strongly depends on subjective perception, and is highly correlated to the specific purpose. There are aspects like the proper semantic level of abstraction, i.e., the granularity of the described workflow actions, which can only be found by an experienced human designer. The notion of appropriateness addressed by this paper rather relates to the control flow perspective and therefore is approachable to measurement but there still remains a subjective element.

The overall aim is to have the model clearly reflect the behavior observed in the log, whereas the degree of appropriateness is determined by both structural properties of the model and the behavior described by it. Figure 1(g) shows *M4*, which is a good model for the event log *L2* as it exactly generates the observed sequences in a structurally suitable way.

In the remainder of this section, both the structural and the behavioral part of appropriateness are considered in more detail.

5.1 Structural Appropriateness

The desire to model a business process in a compact and meaningful way is difficult to capture by measurement. As a first indicator we will define a simple metrics that solely evaluates the size of the graph and subsequently some constructs that may inflate the structure of a process model are considered.

Given the fact that a business process model is expected to have a dedicated *Start* and *End* place, the graph must contain at least one node for every task label, plus two places (the start and end place). Let T be the number of different task labels, and n the number of nodes (i.e., places and transitions) in the Petri net model, then the structural appropriateness metrics a_S is formalized as follows.

$$a_S = \frac{T + 2}{n} \tag{2}$$

Calculating the structural appropriateness for the model *M3* yields $a_S(M3) \approx 0.170$, which is a very bad value caused by the many duplicate tasks. For the good model *M4* the metrics yields $a_S(M4) = 0.5$. With $a_S(M5) \approx 0.435$ a slightly worse value is calculated for the behaviorally (trace) equivalent model *M5* in Figure 1(h), which is now used to consider some constructs that may decrease the structural appropriateness a_S.

(a) *Duplicate tasks.* Duplicate tasks that are used to list alternative execution sequences tend to produce models like the extreme *M3*. *M5*(a) indicates an example situation in which a duplicate task is used to express that after performing activity C either the sequence GH or H alone can be executed. *M4*(b) describes the same process with the help of an invisible task, which is only used for routing purposes and therefore not visible in the log. One could argue that this model supports a more suitable perception namely activity G is not obliged to execute but can be skipped, but it somehow remains a matter of taste. However, excessive usage of duplicate tasks for listing alternative paths reduces the appropriateness of a model in preventing desired abstraction. In addition, there are also duplicate tasks that are necessary to, e.g., specify a certain activity taking place exactly at the beginning and at the end of the process like task A in *M4* (a).

(b) *Invisible tasks*. Besides the invisible tasks used for routing purposes like, e.g., shown in *M4* (b), there are also invisible tasks that only delay visible tasks, such as the one depicted in *M5* (b). If they do not serve any model-related purpose they can simply be removed, thus making the model more concise.

(c) *Implicit places*. Implicit places are places that can be removed without changing the behavior of the model. An example for an implicit place is given in *M5* (c). Again, one could argue that they should be removed as they do not contribute anything, but sometimes it can be useful to insert such an implicit place to, e.g., show document flows. Note that the place *c5* is not implicit as it influences the choice made later on between E and F. Both *c5* and *c10* are *silent places*, with a silent place being a place whose directly preceding transitions are never directly followed by one of their directly succeeding transitions (i.e., the model is unable to produce an event sequence containing BE or AA). Mining techniques by definition are unable to detect implicit places, and have problems detecting silent places.

5.2 Behavioral Appropriateness

Besides the structural properties that can be evaluated on the model itself appropriateness can also be examined with respect to the behavior recorded in the log. Assuming that the log fits the model, i.e., the model allows for all the execution sequences present in the log, there remain those that would fit the model but have not been observed. Assuming further that the log satisfies some notion of completeness, i.e., the behavior observed corresponds to the behavior that should be described by the model, it is desirable to represent it as precisely as possible. When the model gets too general and allows for more behavior than necessary (like in the "flower" model *M2*) it becomes less informative in actually describing the process.

One approach to measure the amount of possible behavior is to determine the mean number of enabled transitions during log replay. This corresponds to the idea that for models clearly reflecting their behavior, i.e., complying with the structural properties mentioned, an increase of alternatives or parallelism and therefore an increase of potential behavior will result in a higher number of enabled transitions during log replay.

Let k be the number of *different* traces from the aggregated log, n the number of process instances combined as one of these traces, m the number of labeled tasks (i.e., does not include invisible tasks, and assuming $m > 1$) in the Petri net model, and x the mean number of enabled transitions during log replay (note that invisible tasks may enable succeeding labeled tasks but they are not counted themselves), then the behavioral appropriateness metrics a_B is formalized as follows.

$$a_B = 1 - \frac{\sum_{i=1}^{k} n_i(x_i - 1)}{(m-1) \cdot \sum_{i=1}^{k} n_i} \tag{3}$$

Calculating the behavioral appropriateness with respect to event log *L2* for the model *M2* yields $a_B(M2, L2) = 0$, which indicates the arbitrary behavior described by it. For *M4*, which exactly allows for the behavior observed in the log, the metrics yields $a_B(M4, L2) \approx 0.967$. As an example it can be compared with the model *M6* in Figure 1(i), which additionally allows for arbitrary loops of activity G and therefore exhibits more potential behavior. This is also reflected by the behavioral appropriateness measure as it yields a slightly smaller value than for the model *M4*, namely $a_B(M6, L2) \approx 0.964$.

5.3 Balancing Fitness and Appropriateness

Having defined the three metrics f, a_S, and a_B, the question is now how to put them together. This is not an easy task since they are partly correlated with each other. So the structure of a process model may influence the fitness metrics f as, e.g., due to inserting redundant invisible tasks the value of f increases because of the more tokens being produced and consumed while having the same amount of missing and remaining ones. But unlike a_S and a_B the metrics f defines an optimal value 1.0, for a log that can be parsed by the model without any error.

Therefore we suggest a conformance testing approach carried out in two phases. During the first phase the fitness of the log and the model is ensured, which means that discrepancies are analyzed and potential corrective actions are undertaken. If there still remain some tolerable deviations, the log or the model should be manually adapted to comply with the ideal or intended behavior, in order to go on with the so-called appropriateness analysis. Within this second phase the degree of suitability of the respective model in representing the process recorded in the log is determined.

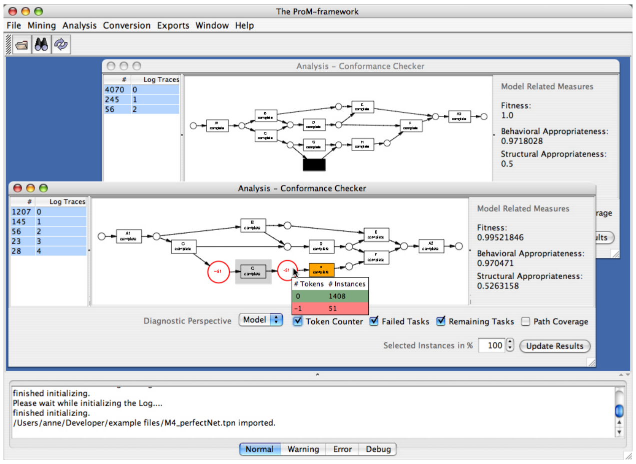
Table 1. Diagnostic results.

	M1	*M2*	*M3*	*M4*	*M5*	*M6*
L1	$f = 1.0$ $a_S = 0.5263$ $a_B = 0.9740$	$f = 1.0$ $a_S = 0.7692$ $a_B = 0.0$	$f = 1.0$ $a_S = 0.1695$ $a_B = 0.9739$	$f = 1.0$ $a_S = 0.5$ $a_B = 0.9718$	$f = 1.0$ $a_S = 0.4348$ $a_B = 0.9749$	$f = 1.0$ $a_S = 0.5556$ $a_B = 0.9703$
L2	$f = 0.9952$ $a_S = 0.5263$ $a_B = 0.9705$	$f = 1.0$ $a_S = 0.7692$ $a_B = 0.0$	$f = 1.0$ $a_S = 0.1695$ $a_B = 0.9745$	$f = 1.0$ $a_S = 0.5$ $a_B = 0.9669$	$f = 1.0$ $a_S = 0.4348$ $a_B = 0.9706$	$f = 1.0$ $a_S = 0.5556$ $a_B = 0.9637$
L3	$f = 0.5397$ $a_S = 0.5263$ $a_B = 0.8909$	$f = 1.0$ $a_S = 0.7692$ $a_B = 0.0$	$f = 0.4947$ $a_S = 0.1695$ $a_B = 0.8798$	$f = 0.6003$ $a_S = 0.5$ $a_B = 0.8904$	$f = 0.6119$ $a_S = 0.4348$ $a_B = 0.9026$	$f = 0.5830$ $a_S = 0.5556$ $a_B = 0.8894$

Regarding the example logs given in Figure 1(b)-(d) this means that we only evaluate the appropriateness measures of those models having a fitness value $f = 1.0$ (cf. Table 1) and therefore completely discard event log *L3*, which only fits the trivial model *M2*, and the model *M1* for event log *L2*. For event log *L1* and *L2* we now want to find the most adequate process model among the remaining ones, respectively. Given the fact that neither the structural appropriateness metrics a_S nor the behavioral appropriateness metrics a_B defines an optimal point (note that for the process model *M2* the a_S value is very high while the a_B value is very low and vice versa for the other extreme model *M3*) they both must be understood as an indicator to be maximized without decreasing the other. A possible outcome of such a qualitative analysis could be that *M1* is selected for *L1* while *M4* is selected for *L2*. More test cases are needed to properly balance the three metrics.

6 Adding Conformance to the ProM Framework

The main concepts discussed in this paper have been implemented in a plug-in for the ProM Framework. The conformance checker replays an event log within a Petri net model in a non-blocking way while gathering diagnostic information that can be accessed afterwards. It calculates the token-based fitness metrics f, taking into account the number of process instances represented by each logical log trace, the structural appropriateness a_S, and the behavioral appropriateness a_B. Furthermore, the diagnostic results can be visualized from both a log-based and model-based perspective.



Analysis – Conformance Checker

#	Log Traces
4070	0
245	1
56	2

Model Related Measures

Fitness:
1.0

Behavioral Appropriateness:
0.9718028

Structural Appropriateness:
0.5

Analysis – Conformance Checker

#	Log Traces
1207	0
145	1
56	2
23	3
28	4

# Tokens	# Instances
0	1408
-1	51

Model Related Measures

Fitness:
0.99521846

Behavioral Appropriateness:
0.970471

Structural Appropriateness:
0.5263158

Diagnostic Perspective Model ☑ Token Counter ☑ Failed Tasks ☑ Remaining Tasks □ Path Coverage

Selected Instances in % 100 Update Results

```
finished initializing.
Please wait while initializing the Log....
finished initializing.
/Users/anne/Developer/example files/M4_perfectNet.tpn imported.
```

Figure 2. Screenshot of the conformance analysis plug-in

During log replay the plug-in takes care of invisible tasks that might enable the transition to be replayed next, and it is able to deal with duplicate tasks. The lower part of Figure 2 shows the result screen of analyzing the conformance of event log *L2* and process model *M1*. As discussed before, for replaying *L2* the model lacks the possibility to skip activity *G*, which also becomes clear in the visualization of the model augmented with diagnostic information. In the other window the process specification *M4* is measured to fit with event log *L1*.

7 Related Work

The work reported in this paper is closely related to earlier work on process mining, i.e., discovering a process model based on some event log. For more information we refer to a special issue of Computers in Industry on process mining [6] and a survey paper [5]. Given the scope of this paper, we are unable to provide a complete listing of the many papers published in recent years.

The work of Cook et al. [8,7] is closely related to this paper. In [8] the concept of process validation is introduced. It assumes an event stream coming from the model and an event stream coming from real-life observations, both streams are compared. Here the time-complexity is problematic as the state-space of the model needs to be explored. In [7] the results are extended to include time aspects. The notion of conformance has also been discussed in the context of security [3], business alignment [1], and genetic mining [13]. However, in each of the papers mentioned only fitness is considered and appropriateness is mostly ignored. In [10] the process mining problem is faced with the aim of deriving a model which is as compliant as possible with the log data, accounting for fitness (called completeness) and also behavioral appropriateness (called soundness).

Process mining and conformance testing can be seen in the broader context of Business (Process) Intelligence (BPI) and Business Activity Monitoring (BAM). Tools such as described in [11,14], however, often focus on performance measurements rather than monitoring (un)desirable behavior.

8 Conclusion

Given the presence of both process models and event logs in most organizations of some complexity, it is interesting to investigate the notion of conformance as it has been defined in this paper. Conformance is an important notion in the context of business alignment, auditing (cf. Sarbanes-Oxley (SOX) Act [15]), and business process improvement. Therefore, the question "Does the event log *conform* to the process model and vice versa?" is highly relevant.

We have shown that conformance has two dimensions: fitness and appropriateness. Fitness can be captured in one metrics (f). For measuring appropriateness we introduced two metrics: structural appropriateness a_S and behavioral appropriateness a_B. Together these three metrics allow for the quantification of conformance. The metrics defined in this paper are supported by the *Conformance Checker*, a tool which has been implemented within the ProM Framework.

An interesting direction for future research is to exploit log data on a more fine-grained level (e.g., stating the *start* and *end* of activities) and to include other perspectives such as time, data, and resources. For example, in some application the timing of an event is as important as its occurrence.

Acknowledgements
The authors would like to thank Ton Weijters, Boudewijn van Dongen, Ana Karla Alves de Medeiros, Minseok Song, Laura Maruster, Eric Verbeek, Monique Jansen-Vullers, Hajo Reijers, Michael Rosemann, Huub de Beer, Peter van den Brand, et al. for their on-going work on process mining techniques.

References

1. W.M.P. van der Aalst. Business Alignment: Using Process Mining as a Tool for Delta Analysis. In J. Grundspenkis and M. Kirikova, editors, *Proceedings of the 5th Workshop on Business Process Modeling, Development and Support (BPMDS'04)*, volume 2 of *Caise'04 Workshops*, pages 138–145. Riga Technical University, 2004.
2. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
3. W.M.P. van der Aalst and A.K.A. de Medeiros. Process Mining and Security: Detecting Anomalous Process Executions and Checking Process Conformance. In N. Busi, R. Gorrieri, and F. Martinelli, editors, *Second International Workshop on Security Issues with Petri Nets and other Computational Models (WISP 2004)*, pages 69–84. STAR, Servizio Tipografico Area della Ricerca, CNR Pisa, Italy, 2004.
4. W.M.P. van der Aalst and M. Song. Mining Social Networks: Uncovering Interaction Patterns in Business Processes. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 244–260. Springer-Verlag, 2004.
5. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
6. W.M.P. van der Aalst and A.J.M.M. Weijters, editors. *Process Mining*, Special Issue of Computers in Industry, Volume 53, Number 3. Elsevier Science Publishers, Amsterdam, 2004.
7. J.E. Cook, C. He, and C. Ma. Measuring Behavioral Correspondence to a Timed Concurrent Model. In *Proceedings of the 2001 International Conference on Software Mainenance*, pages 332–341, 2001.
8. J.E. Cook and A.L. Wolf. Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. *ACM Transactions on Software Engineering and Methodology*, 8(2):147–176, 1999.
9. J. Desel, W. Reisig, and G. Rozenberg, editors. *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.
10. G. Greco, A. Guzzo, L. Pontieri, and D. Saccá. Mining Expressive Process Models by Clustering Workflow Traces. In *Proc of Advances in Kowledge Discovery and Data Mining, 8th Pacific-Asia Conference (PAKDD 2004)*, pages 52–62, 2004.
11. D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.C. Shan. Business process intelligence. *Computers in Industry*, 53(3):321–343, 2004.
12. G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley, Reading MA, 1998.
13. A.K.A. de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. Using Genetic Algorithms to Mine Process Models: Representation, Operators and Results. BETA Working Paper Series, WP 124, Eindhoven University of Technology, 2004.
14. M. zur Mühlen and M. Rosemann. Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues. In R. Sprague, editor, *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33)*, pages 1–10. IEEE Computer Society Press, Los Alamitos, California, 2000.
15. P. Sarbanes, G. Oxley, and et al. Sarbanes-Oxley Act of 2002, 2002.
16. A.W. Scheer. *ARIS: Business Process Modelling*. Springer-Verlag, Berlin, 2000.