Business Process Performance Mining with Staged Process Flows

Hoang Nguyen[1], Marlon Dumas[2], Arthur H.M. ter Hofstede[1], Marcello La Rosa[1], and Fabrizio Maria Maggi[2]

[1] Queensland University of Technology, Australia
huanghuy.nguyen@hdr.qut.edu.au,
{a.terhofstede,m.larosa}@qut.edu.au
[2] University of Tartu, Estonia
{marlon.dumas,f.m.maggi}@ut.ee

Abstract. Existing business process performance mining tools offer various summary views of the performance of a process over a given period of time, allowing analysts to identify bottlenecks and their performance effects. However, these tools are not designed to help analysts understand how bottlenecks form and dissolve over time nor how the formation and dissolution of bottlenecks – and associated fluctuations in demand and capacity – affect the overall process performance. This paper presents an approach to analyze the evolution of process performance via a notion of Staged Process Flow (SPF). An SPF abstracts a business process as a series of queues corresponding to stages. The paper defines a number of stage characteristics and visualizations that collectively allow process performance evolution to be analyzed from multiple perspectives. It demonstrates the advantages of the SPF approach over state-of-the-art process performance mining tools using a real-life event log of a Dutch bank.

Keywords: Process mining, performance analysis, multistage processes, cumulative flow, queuing theory.

1 Introduction

Process mining is a family of techniques designed to extract insights from business process event logs [1]. *Process Performance Mining* (PPM) is a subset of process mining techniques concerned with the analysis of processes with respect to performance dimensions, chiefly *time* (how fast a process is executed); *cost* (how much a process execution costs); *quality* (how well the process meets customer requirements and expectations); and *flexibility* (how rapidly can a process adjust to changes in the environment) [2].

Along the time and flexibility dimensions, one recurrent analysis task is to understand how the temporal performance of a process evolves over a given period of time – also known as *flow performance* analysis in lean management [3]. For example, a bank manager may wish to know how the waiting times in a loan application process have evolved over the past month in order to adjust the resource allocation policies so as to minimize the effects of bottlenecks.

Existing PPM techniques are not designed to address such flow performance questions. Instead, these techniques focus on analyzing process performance in a "snapshot" manner, by taking as input an event log recorded during a period of time and extracting aggregate measures such as mean waiting time, processing time or cycle time of the process and its activities. For example, both the Performance Analysis plugins of ProM [4] and Disco [5] calculate aggregate performance measures (e.g. mean waiting time) over the entire period covered by an event log and display these measures by color-coding the elements of a process model. These tools can also produce animations of the flow of cases along a process model over time. However, extracting flow performance insights from these animations requires close and continuous attention from the analyst in order to detect visual cues of performance trends, bottleneck formation and dissolution, and phase transitions in the process performance. In other words, animation techniques allow analysts to get a broad picture of performance issues, but not to precisely quantify the evolution of process performance over time.

In this setting, this paper presents a PPM approach designed to provide a precise and quantifiable picture of flow performance. The approach relies on an abstraction of business processes called *Staged Process Flow* (SPF). An SPF breaks down a process into a series of queues corresponding to user-defined stages. Each stage is associated with a number of performance characteristics that are computed at each time point in an observation window. The evolution of these characteristics is then plotted via several visualization techniques that collectively allow flow performance to be analyzed from multiple perspectives in order to address the following questions:

Q1. How does the overall process performance evolve over time?
Q2. How does the formation and dissolution of bottlenecks affect the overall process performance?
Q3. How do changes in demand and capacity affect the overall process performance?

The rest of this paper is organized as follows. Section 2 reviews existing PPM techniques with respect to the problem of flow performance analysis. Section 3 describes the SPF concept and associated characteristics and visualizations. Section 4 discusses an evaluation of the approach based on a real-life log. Finally, Section 5 summarizes the contributions and outlines future work directions.

2 Related Work

Existing PPM tools support the analysis of entire processes or activities thereof with respect to performance measures such as cycle time, processing time and waiting time. Some PPM tools display the distribution of performance measures in the form of dashboards (e.g. bar charts) alongside aggregate statistics (e.g. mean and median) [5]. Others overlay the performance measures on top of a process model, for example by replaying the log on the process model [4, 6] and calculating aggregate performance measures for each element in the process model during replay. Techniques for enhancing the quality of log replaying based on clustering techniques have been proposed [7]. All these techniques are designed to summarize the performance of the process over the entire time period covered by the event log. They can pinpoint bottlenecks, resource underutilization and other performance issues observed across said time period. However, they do not allow one to analyze how those bottlenecks form and dissolve, and more generally, how the performance of the process varies over time.

There is a range of techniques to extract and analyze process performance characteristics (incl. performance measures) from event logs. For example, de Leoni et al. [8] propose a framework to extract process performance characteristics from event logs and to correlate them in order to discriminate for example between the performance of cases that lead to “positive” outcomes versus “negative” outcomes. Meanwhile, Pika et al. [9] propose a framework to extract performance characteristics along the resource perspective. These proposals however are not designed to provide insights into the evolution of process performance over time.

A related technique supported by contemporary PPM tools is log animation. Log animation displays in a movie-like fashion how cases circulate through the process model over time [10, 7, 11]. However, extracting flow performance insights from these animations requires the analyst to: (i) manually look for visual cues in the animation that indicate trends, phase transitions or bottlenecks in the process' performance; and (ii) run additional queries to locate and quantify the observed performance phenomena.

Process performance has also been approached from the perspective of queuing theory. Senderovich et al. [12] propose a method to discover characteristics of "work queues" from event logs at the level of an entire process or of individual activities. Meanwhile, Smet [13] proposes a method to discover collections of queues from event logs. This latter method discovers queues by grouping resources and activities into clusters based on cohesion metrics. The queuing models produced by the above methods are used for prediction (e.g. of waiting times) rather than performance analysis. As such these methods are only marginally related to the problem of flow performance analysis.

The concept of SPF presented in this paper is inspired by flow performance analysis techniques from the fields of lean management and agile software engineering. The idea of decomposing the process into stages and analyzing flow metrics at each stage can be found in various embodiments in contemporary lean and agile management tools, e.g. Kanban Flow[1] and ActionableAgile[2]. The concept of SPF formalized in this paper in the context of business process event logs, provides a generic framework that brings together flow performance analysis techniques found across these tools.

3 Approach

In this section, we introduce the concept of SPF and its formalization before describing our SPF-based approach to process performance mining.

3.1 SPF overview

An SPF is a partitioning of the set of log events into consecutive *stages* with a defined order (e.g. $\langle s_1, s_2, s_3, s_4 \rangle$). For each trace, all events in one stage must precede all events in the subsequent stage (in our example all events in s_1 must have a causal relation with all events in s_2). A trace does not need to have all stages, so long as its stages follow the defined order (in our example a trace with $\langle s_1, s_2 \rangle$ is possible but not a trace with $\langle s_1, s_2, s_4 \rangle$). We model a stage as a *queuing system*, where the queuing items are cases and the service facility is the set of resources available to handle cases in the stage in question. Each stage has an *arrival flow* via which new cases arrive to the stage in question and a *departure flow* via which cases depart. In addition, a stage may have *exit flows*, capturing the fact that a case may leave the process abnormally after being serviced at a stage. This will be the cases for traces that do not finish all stages.

For illustration, we use the loan origination process of a Dutch bank, which was the subject of the BPI Challenge 2012 log.[3] As depicted in the SPF in Fig. 1, a case in this process is a loan application that goes through four stages: Pre-Assess (s_1), Assess (s_2), Negotiate (s_3) and Validate (s_4), in this order. In the "Pre-Assess" stage, the bank checks the completeness of the loan application and requests the customer to provide sufficient documents before their application can proceed to the next stage.

[1] `http://kanbanflow.com`
[2] `http://www.actionableagile.com`
[3] `http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`

Next, in the "Assess" stage, the bank checks the eligibility of the loan application. In the "Negotiate" stage, the bank and the customer discuss the terms and conditions of the loan until it is ready for validation. Finally, in the "Validate" stage, a bank controller reviews and decides to approve or reject the loan application. At the end of any stage, a loan application can either be declined by the bank or canceled by the customer, which leads to interrupting the process at that point.

In this example, each stage has an exit flow consisting of loan applications that are declined or canceled. Thus, a trace recording a loan application that is canceled after the assessment, will only have the first two stages.

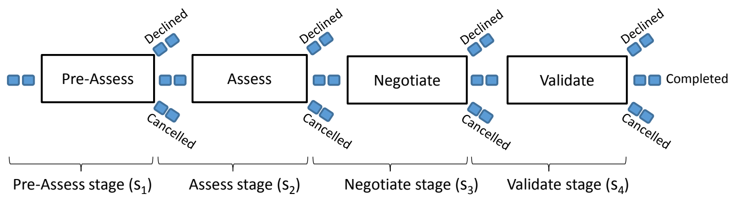


Fig. 1. SPF model of a loan origination process.

Flow performance in an SPF is determined by a set of *characteristics* capturing the interplay between the arrival flow on the one hand and the departure and exit flows on the other. One such characteristic is the *Cases In Progress* (in reference to "Work-in-Progress"), that is, the set of cases found in a stage at a given point in time. Another characteristic is the *Time in Stage*: the time between the arrival and the departure/exit of a case for a given stage. Each case spends a certain amount of time waiting in a stage, and another amount of time being processed in that stage. *Flow Efficiency* is the ratio between the processing time of a case in a stage and the Time in Stage. Below we formally define how an SPF and its characteristics are extracted from an event log.

3.2 SPF formalization

An event log is the starting point of any process mining task. Fig. 2 shows an event log of a loan origination process. An event log consists of a set of *cases*, where a case is a uniquely identified instance of a process. For example, the loan application identified by code c_4 is a case. Each case consists of a sequence of *events*. An event is the most granular element of a log and is characterized by a set of attributes such as *activity*, *resource* (the entity that performed the activity associated with the event, which can be human or non-human), and *timestamp* (the moment when the event occurred). *Event type* represents the association between an event and its activity's lifecycle, such as "schedule", "start", and "complete". In this paper, we assume that "start" and "complete" are the only event types associated with activities.

Formally, an event log EL is a tuple $(E, ET, A, R, C, time, act, type, res, case)$, where E is a set of events, $ET = \{start, complete\}$ is the set of event types, A is a set of globally (i.e. across cases) unique activity identifiers (AID), R is a set of resources, C is a set of cases, $time : E \to \mathbb{R}_0^+$ is a function that assigns a timestamp to an event, $act : E \to A$ is a function that assigns an AID to an event, $type : E \to ET$ is a function that assigns an event type to an event, $res : E \to R$ is a function that assigns a resource to an event, and $case : E \to C$ relates an event to a case. We write $e \lesssim_E e'$ iff $time(e) \leq time(e')$.

In our model, events are associated with *stages*. For example, a particular "Check application" event occurs at the "Assess" stage of a loan application. A completed case is one that passed all stages and has a "complete" status, otherwise, the case is considered to have exited the process prematurely and will have the status "incomplete".

Case ID	Case Status	Reason	Stage	Event ID	Event Type	Timestamp	Activity Name	AID	Res.
c_1	Incomplete	Declined	Pre-Assess (S_1)	e_1	start	05.10 09:00:00	Update application	a_1	Rob
				e_2	complete	05.10 10:00:00	Update application	a_1	Rob
c_2	Incomplete	Declined	Pre-Assess (S_1)	e_3	start	06.10 09:00:00	Update application	a_2	Rob
				e_4	complete	06.10 10:00:00	Update application	a_2	Rob
			Assess (S_2)	e_5	start	08.10 09:00:00	Check application	a_3	Sara
				e_6	complete	08.10 10:00:00	Check application	a_3	Sara
				e_7	start	09.10 08:30:00	Check application	a_4	Sara
				e_8	complete	09.10 09:00:00	Check application	a_4	Sara
c_3	Incomplete	Canceled	Pre-Assess (S_1)	e_9	start	08.10 09:00:00	Update application	a_5	Rob
				e_{10}	complete	08.10 10:00:00	Update application	a_5	Rob
			Assess (S_2)	e_{11}	start	09.10 09:00:00	Check application	a_6	Sara
				e_{12}	complete	09.10 09:15:00	Check application	a_6	Sara
			Negotiate (S_3)	e_{13}	start	11.10 09:00:00	Follow up offer	a_7	Sara
				e_{14}	complete	11.10 10:00:00	Follow up offer	a_7	Sara
c_4	Complete		Pre-Assess (S_1)	e_{15}	start	09.10 08:00:00	Update application	a_8	Rob
				e_{16}	complete	09.10 09:00:00	Update application	a_8	Rob
			Assess (S_2)	e_{17}	start	09.10 09:00:00	Check application	a_9	Tim
				e_{18}	complete	09.10 10:00:00	Check application	a_9	Tim
			Negotiate (S_3)	e_{19}	start	10.10 09:00:00	Follow up offer	a_{10}	Tim
				e_{20}	complete	10.10 10:00:00	Follow up offer	a_{10}	Tim
			Validate (S_4)	e_{21}	start	12.10 09:00:00	Validate application	a_{11}	Mike
				e_{22}	complete	12.10 10:00:00	Validate application	a_{11}	Mike

Fig. 2. Example event log for a loan origination process.

Stage-based enhancement In our approach, an event log must firstly be enhanced with stage information. A *stage-based enhancement SE* of an event log $EL = (E, ET, A, R, C, time, act, type, res, case)$ is defined as a tuple $(S, CS, <_S, stage, status)$, where S is a set of stages, $CS = \{complete, incomplete\}$ a set of case statuses, $<_S \subseteq S \times S$ a strict total order over S (with $\leqslant_S$ the corresponding total order), $stage : E \to S$ assigns stages to events, and $status : C \to CS$ assigns statuses to cases. For convenience, we write $E^{c,s} = \{e \in E \mid case(e) = c \wedge stage(e) = s\}$ to denote the set of all events of case c that occurred in stage s, and $E^{start} = \{e \in E \mid type(e) = start\}$ is the set of all "start" events.

While there can be a number of ways to arrive at a stage-based enhancement of an event log, there are a number of rules that need to be satisfied. First of all, if a case covers a stage s, i.e. there is at least one event belonging to that stage, there must be events associated with all stages preceding s in that case:

$$\forall c \in C\ \forall s \in S[E^{c,s} \neq \varnothing \Rightarrow \forall s' \in S[s' <_S s \Rightarrow E^{c,s'} \neq \varnothing]].$$

The stages covered by a case must observe the defined order $<_S$ over S:

$$\forall e, e' \in E[(case(e) = case(e') \wedge e \lesssim_E e') \Rightarrow stage(e) \leqslant_S stage(e')].$$

Events related to the same activity must belong to the same stage:

$$\forall e, e' \in E[act(e) = act(e') \Rightarrow stage(e) = stage(e')].$$

If a case has a complete status, it should have gone through all the stages:

$$\forall c \in C[status(c) = complete \Rightarrow \forall s \in S\ \exists e \in E[case(e) = c \wedge stage(e) = s]].$$

SPF characteristics The start of stage s in case c, $T_{AR}(c, s)$, is defined as $\min\limits_{e \in E^{c,s}} time(e)$ if $E^{c,s} \neq \varnothing$ and is undefined ($\perp$) otherwise. Similarly, the end of a stage s in case c, $T_{DP}(c, s)$, is $\max\limits_{e \in E^{c,s}} time(e)$ if $E^{c,s} \neq \varnothing$ and is undefined otherwise.

For all timestamps t neither $t < \perp$ nor $t > \perp$ holds. The last stage of case c is s, $laststage(c, s)$, iff $\neg(\exists s' \in S\, \exists e \in E[s <_S s' \wedge case(e) = c \wedge stage(e) = s'])$.

The set $C_{AR}(s, t)$ consists of all cases that have reached stage s on or before t, i.e. $C_{AR}(s, t) \triangleq \{c \in C \mid \exists e \in E^{c,s}[time(e) \leq t]\}$. Similarly, the set $C_{DP}(s, t)$ consists of all cases that have gone beyond stage s on or before time t, i.e. $C_{DP}(s, t) \triangleq \{c \in C \mid \forall e \in E^{c,s}[time(e) \leq t]\}$. The set $C_{EX}(s, t)$ consists of those cases that have completed stage s on or before time t, have not gone beyond stage s, and are considered to be incomplete: $C_{EX}(s, t) \triangleq \{c \in C \mid \forall e \in E^{c,s}[time(e) \leq t] \wedge laststage(c, s) \wedge status(c) = incomplete\}$.

The *Arrival/Departure/Exit Rate* X (X stands for AR, DP, and EX) is the average number of cases arriving at/departing from/exiting after a stage s per unit of time Δ at a given point in time t:

$$X(s, t, \Delta) \triangleq \frac{|C_X(s, t)| - |C_X(s, t - \Delta)|}{\Delta}.$$

It is required that $\Delta > 0$ here and elsewhere, and $t - \Delta$ is not before all case start times in the log, i.e. $\exists e \in E[time(e) \leq (t - \Delta)]$.

Cases in Progress is the number of cases present at a stage s at a point in time t:

$$CIP(s, t) \triangleq |C_{AR}(s, t)| - |C_{DP}(s, t)|.$$

The *Time in Stage* for a point in time t and a stage s is the minimal duration that one needs to wait to see the number of departing cases from s equal or greater than the number of cases that arrived in stage s on or before time t. Formally, let t' be the minimal timestamp such that $t' = t + i\Delta (i = 1 \ldots n)$ and $|C_{DP}(s, t')| \geq |C_{AR}(s, t)|$, then $TIS(s, t, \Delta) = t' - t$. $TIS(s, t, \Delta)$ is undefined if no such t' exists.

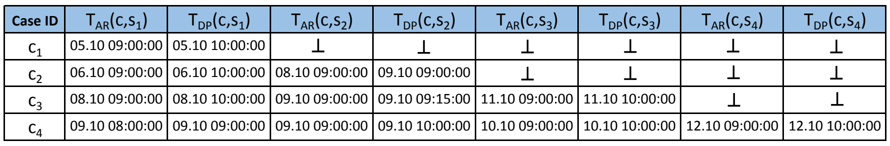
Finally, the *Flow Efficiency FE* of a stage s during an interval $[t - \Delta, t]$ is the sum of all durations of activities that occurred in that stage and that interval divided by the sum of all case durations for that stage in the said interval. To be able to determine the durations of activities, we have to impose further requirements on an event log: (1) for every activity in the log there is at most one corresponding "start" event and one corresponding "complete" event, i.e. $\forall e \in E\, \nexists e' \in E[e \neq e' \wedge act(e') = act(e) \wedge type(e') = type(e)]$, (2) for every "start" event of an activity there is a corresponding "complete" event and vice versa, i.e. $\forall e \in E\, \exists e' \in E[act(e) = act(e') \wedge type(e) \neq type(e')]$, and (3) for every activity its corresponding "start" event should occur before its corresponding "complete" event, i.e. $\forall e \in E\, \forall e' \in E[(type(e) = start \wedge type(e') = complete \wedge act(e) = act(e')) \Rightarrow time(e) < time(e')]$. Then, any activity $a \in ran(act)$ has exactly one corresponding "start" event e_s and exactly one corresponding "complete" event e_c. The duration of a during a closed time interval $[t_1, t_2]$, denoted $dur(a, t_1, t_2)$, is defined as $[t_1, t_2] \cap [time(e_s), time(e_c)]$. In addition, the duration of case c at stage s within interval $[t_1, t_2]$, written $dur(c, s, t_1, t_2)$, is defined as $[t_1, t_2] \cap [T_{AR}(c, s), T_{DP}(c, s)]$ if $E^{c,s} \neq \varnothing$ and is zero (0) otherwise.

$$FE(s, t, \Delta) \triangleq \frac{\sum\limits_{e \in E^{start}, stage(e) = s} dur(act(e), t - \Delta, t)}{\sum\limits_{c \in C} dur(c, s, t - \Delta, t)}.$$

Note that at least one case should have events in the interval $[t - \Delta, t]$ in stage s to avoid the denominator evaluating to zero.

The formulae above can be illustrated with the example log given in Fig. 2. First, the log is summarized by stages and cases as shown in Fig. 3 for ease of computation. With $t_1 = $ 09.10 08:15:00, $t_2 = $ 09.10 09:15:00, $\Delta = 1h$, the values of the SPF characteristics are computed as follows:

- $C_{AR}(s_2, t_1) = \{c_2\}$, $C_{AR}(s_2, t_2) = \{c_2, c_3, c_4\}$
- $C_{DP}(s_2, t_1) = \{\}$, $C_{DP}(s_2, t_2) = \{c_2, c_3\}$
- $C_{EX}(s_2, t_1) = \{\}$, $C_{EX}(s_2, t_2) = \{c_2\}$
- $AR(s_2, t_2, \Delta) = 2$ cases/h, $DP(s_2, t_2, \Delta) = 2$ cases/h, $EX(s_2, t_2, \Delta) = 1$ case/h
- $CIP(s_2, t_2) = 1$ case, $TIS(s_2, t_2, \Delta) = 1$h (at $t' = $ 09.10 10:15:00, $C_{DP}(s_2, t') = C_{AR}(s_2, t_2)$)
- $FE(s_2, t_2, \Delta) = \frac{dur(a_3,t_1,t_2)+dur(a_4,t_1,t_2)+dur(a_6,t_1,t_2)+dur(a_9,t_1,t_2)}{dur(c_1,s_2,t_1,t_2)+dur(c_2,s_2,t_1,t_2)+dur(c_3,s_2,t_1,t_2)+dur(c_4,s_2,t_1,t_2)} = \frac{0+30min+15min+15min}{0+45min+15min+15min} = 0.8.$



Case ID	$T_{AR}(c,s_1)$	$T_{DP}(c,s_1)$	$T_{AR}(c,s_2)$	$T_{DP}(c,s_2)$	$T_{AR}(c,s_3)$	$T_{DP}(c,s_3)$	$T_{AR}(c,s_4)$	$T_{DP}(c,s_4)$
c_1	05.10 09:00:00	05.10 10:00:00	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
c_2	06.10 09:00:00	06.10 10:00:00	08.10 09:00:00	09.10 09:00:00	$\perp$	$\perp$	$\perp$	$\perp$
c_3	08.10 09:00:00	08.10 10:00:00	09.10 09:00:00	09.10 09:15:00	11.10 09:00:00	11.10 10:00:00	$\perp$	$\perp$
c_4	09.10 08:00:00	09.10 09:00:00	09.10 09:00:00	09.10 10:00:00	10.10 09:00:00	10.10 10:00:00	12.10 09:00:00	12.10 10:00:00

Fig. 3. Stage-based timetable.

3.3 SPF-based performance mining approach

Our approach to process performance mining follows three steps: i) construct flow cells; ii) measure SPF characteristics; iii) visualize SPF characteristics for user consumption.

Construct flow cells First, the log is enhanced with stage information. This is currently done via preprocessing, which consists in adding two stage-based attributes: a "stage" attribute for each event, indicating which stage it belongs to, and a "status" attribute to the case, indicating if the case is complete.

Next, the timeline of the log is divided into equal time intervals Δ. The stages and time intervals create a two-dimensional space (see Fig. 4), in which a cell at the intersection of a stage and an interval located at $t_i = t_o + i\Delta$ $(i = 0 \ldots n$, and t_o is the starting time of the log) is called a *flow cell*. From the stage-based timetable (e.g. Fig. 3), it is possible to check exactly which flow cells a case falls in during its lifecycle.

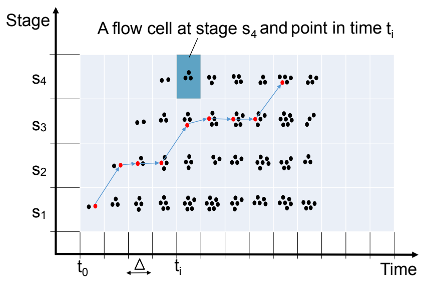


Fig. 4. Flow cells.

Measure SPF characteristics SPF characteristics are computed first at every flow cell, then rolled up to the stage and process level (also called system level). At a particular flow cell located at a stage s and a point in time $t_i = t_o + i\Delta$ $(i = 0 \ldots n)$, the formulae presented in Section 3.2 can be applied as exemplified above. At a stage s, the SPF characteristics of s for a time interval are computed as a statistic (e.g. max, min, and mean) of the corresponding SPF characteristics of all flow cells located at stage s and fully contained within the time interval. Similarly, at the system level, the SPF characteristics of the system are computed as a statistic of the corresponding characteristics of all stages, except that the Arrival Rate and Departure Rate at the system level are the Arrival Rate at the first stage and Departure Rate at the last stage, respectively.

Visualize SPF characteristics Based on the above formalization, we provide three visualizations to support the analysis of SPF characteristics at different levels of abstraction and periods of time.

Cumulative Flow Diagram (CFD): A CFD is an area graph used in queueing theory [14] to visualize the evolution of flow performance over time. Fig. 5 depicts how some SPF characteristics are related to the geometry of the CFD. In our case, each area, encoded with a different color, represents the number of cases queuing for a given process stage (*queue flow*), being worked in that stage (*service flow*) or exiting from that stage (*exit flow*). The service flow and the queue flow are actually two sub-stages of a process stage with similar SPF characteristics. The CFD is particularly suitable for examining the flow performance. For example, one can observe the process evolution over time through the development trend of different flows, identify the formation and dissolution of a bottleneck through widening and shrinking areas on a queue and service flow, and detect patterns of changes in the arrival rate and departure rate as well as their correlation with the process performance.

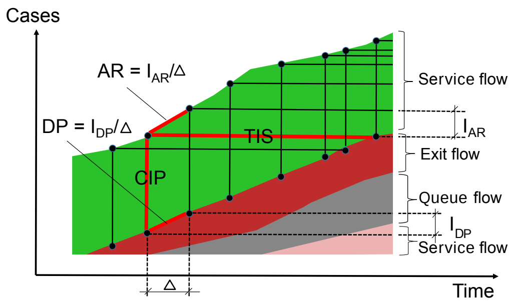
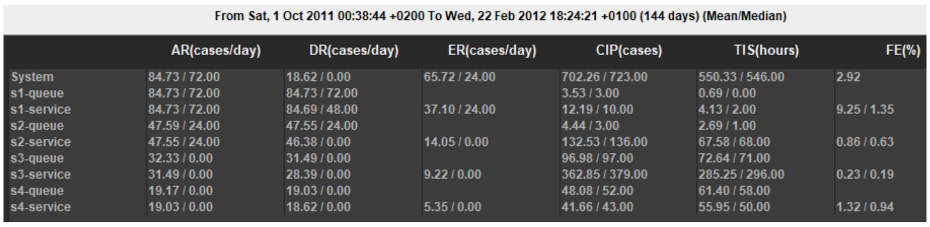


Fig. 5. CFD structure.

Performance Summary Table (PST): the PST (Fig. 6) provides a quick and exact measurement of the flow performance in figures, at the stage and system levels. It also allows one to measure the flow for any time interval of the log.


From Sat, 1 Oct 2011 00:38:44 +0200 To Wed, 22 Feb 2012 18:24:21 +0100 (144 days) (Mean/Median)

	AR(cases/day)	DR(cases/day)	ER(cases/day)	CIP(cases)	TIS(hours)	FE(%)
System	84.73 / 72.00	18.62 / 0.00	65.72 / 24.00	702.26 / 723.00	550.33 / 546.00	2.92
s1-queue	84.73 / 72.00	84.73 / 72.00		3.53 / 3.00	0.69 / 0.00	
s1-service	84.73 / 72.00	84.69 / 48.00	37.10 / 24.00	12.19 / 10.00	4.13 / 2.00	9.25 / 1.35
s2-queue	47.59 / 24.00	47.55 / 24.00		4.44 / 3.00	2.69 / 1.00	
s2-service	47.55 / 24.00	46.38 / 0.00	14.05 / 0.00	132.53 / 136.00	67.58 / 68.00	0.86 / 0.63
s3-queue	32.33 / 0.00	31.49 / 0.00		96.98 / 97.00	72.64 / 71.00	
s3-service	31.49 / 0.00	28.39 / 0.00	9.22 / 0.00	362.85 / 379.00	285.25 / 296.00	0.23 / 0.19
s4-queue	19.17 / 0.00	19.03 / 0.00		48.08 / 52.00	61.40 / 58.00	
s4-service	19.03 / 0.00	18.62 / 0.00	5.35 / 0.00	41.66 / 43.00	55.95 / 50.00	1.32 / 0.94

Fig. 6. Performance Summary Table (AR=Arrival Rate, DR=Departure Rate, ER=Exit Rate, CIP=Cases in Progress, TIS=Time in Stage, FE=Flow Efficiency). For example, AR=84.73/72.00 indicates that the mean arrival rate is 84.73 cases per day and the median rate is 72 cases per day.

Time Series Charts (TSCs): As most SPF characteristics are time-dependent, TSCs (Fig. 7) can be used to investigate the evolution of SPF stage characteristics over time, such as viewing the development of arrival rate, the difference between departure and arrival rate at different intervals, or the formation of bottlenecks over time. Fig. 7 gives a multiple-series TSC showing the evolution of various SPF characteristics over time.

4 Evaluation

We implemented our approach as a ProM plugin, namely the "Performance Mining With Staged Process Flows" plugin, as well as a standalone Java application.[4] In the

[4] Available from `http://promtools.org` (ProM) and `http://apromore.org/platform/tools` (standalone Java application).

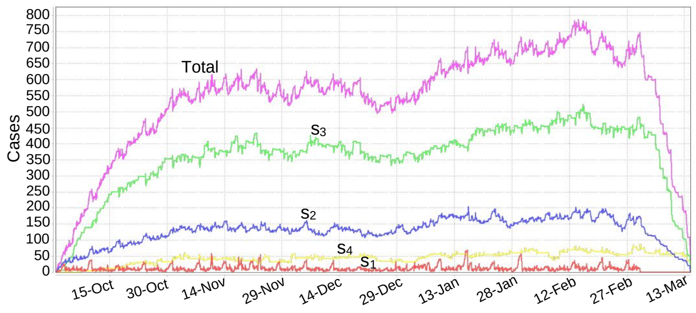


Fig. 7. Time Series Chart of various SPF characteristics.

following, we use this implementation to answer the questions raised in Section 1 using the BPI Challenge 2012 log, and compare the results with those obtained from two state-of-the-art PPM tools. For space reasons, the results of a second evaluation, using the BPI Challenge 2013 log,[5] are provided in a technical report [15], though they are in line with those reported in this paper.

The BPI Challenge 2012 log records cases of a loan origination process at a Dutch bank (see Section 3 for a description). It contains 13,087 loan applications with a total of 193,208 events occurring from 1 Oct 2011 to 15 Mar 2012. Every case must pass four stages. The completion of each stage is marked by a special event such as A_PREACCEPTED, A_ACCEPTED, and A_ACTIVATED. We preprocessed this log to enhance it with stage information, including adding a "stage" attribute for events and a "status" attribute for cases.

The PPM tools evaluated are the "Performance Analysis with Petri Net" plugin of ProM 5.2 [4] (PEP for short), and Fluxicon's Disco [5]. PEP requires a Petri net discovered from an event log as input. The net can be obtained by using any of the available discovery algorithms of ProM that either directly discovers a Petri net or whose result can be converted into a Petri net, such as the Heuristics Miner. PEP can be run to internally replay the log on the Petri net, in order to align the log with the model, compute time-related performance information and overlay it to the model. Specifically, processing time is assigned to Petri net transitions (capturing process activities) while waiting time is assigned to places (capturing states). Arrival rates for these elements are also provided. Moreover, places are color-coded based on the length of the waiting time (blue for short waits, yellow for medium and red for long). The thresholds for the colors can be set automatically or manually by the user. The tool also provides overall performance measures such as arrival rate and statistics on cycle time.[6]

Similar to PEP, Disco's performance measurements are mainly based on a process model. The tool takes an event log as input and discovers a Fuzzy net, which provides an abstract representation of the process behavior, by showing the process activities and paths connecting these activities. This model is enhanced with frequency information and statistics on performance measures at the level of individual process activities (processing time) and paths (waiting time). The complexity of the discovered model can be adjusted based on case frequency, in order to obtain a simpler process model that abstracts away infrequent cases. Different types of filters besides frequency can be used to create model projections which can be used to compare process variants on the basis of

[5] `http://dx.doi.org/10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07`

[6] The "Replay a Log on Petri Net for Performance/Conformance Analysis" plugin of ProM 6 works in a similar way to PEP, though it provides less performance information.

their performance, e.g. focusing on all cases that have a duration or a number of events within a given range. In addition, Disco can replay the log on the discovered model.

For each question, we evaluated each tool along the quality dimensions of *ease of use* and *usefulness*, widely used in technology acceptance models [16]. In our context, ease of use refers to the effort required from the user to retrieve and to interpret data in order to answer a given question. Usefulness on the other hand refers to the extent the tool provides data that allows the user to answer the question in a precise (i.e. quantitatively) and informative manner. Below we evaluate the three tools for each question.

Q1: How does the overall process performance evolve over time?

SPF The evolution of the process is depicted on a CFD (Fig. 8). The shape of the CFD reflects the development of the process at each stage. The characteristics, such as arrival rate (AR) and cases in progress (CIP), can be seen at any point in time as a tooltip. The CFD can be zoomed in to investigate patterns of evolution at different intervals (e.g. weekly, daily and hourly). The evolution can also be viewed on the plot of flow efficiency over time. The PST (Fig. 6) provides a summary of the flow performance at any time interval. From these visualizations, we can draw the following observations:

- The process has a stable trend indicated through the even height of service flows shown in Fig. 8 (bands named as s_i-Service). Further evidence is provided by the average arrival and departure rates, which are comparable at each stage in Fig. 6, and by the fact that there is little variation between the average mean and median value of CIP and TIS.
- There are strong exit flows throughout the period from s_1 (strongest) to s_4 (bands named as s_i-Exit on the CFD). Apparently, these exit flows contribute to keeping the arrival of cases at each stage on a par with their departure.
- The CFD and PST show that the waiting queue is negligible at stage s_1 but starts to emerge at stage s_2 and becomes considerable at stages s_3 and s_4, meaning that the process has slower response in the later stages.
- The process has very low flow efficiency (3%), i.e. 97% of time a case stays idle. The problem seems to be with frequent waits for customer response.

As shown above, the SPF proposes an easy way to understand how the overall process performance evolves over time. The output is easy to interpret, as it is based on visual cues and performance measures; precise, as it is supported by numeric measurements; and most importantly, it leads to various insightful observations.

PEP An excerpt of the Petri net enhanced with performance information provided by PEP is shown in Fig. 9. This model was obtained by first discovering a Heuristics net from the log and then converting it to a Petri net. However, in order to obtain a model that is easy to interpret, we had to incrementally filter the log, as the first model discovered was a spaghetti-like model too dense to understand. Eventually, we ended up retaining only those events that mark the end of each stage in the log (i.e. the "gate" events), in a similar vein to our approach. A drawback of this operation is that the fitness of the model decreases as some traces of the log can no longer be replayed on the model. As a result, the performance measures provided by the tool are only approximate, as they only refer to those traces that perfectly fit the model.

Coming back to Q1, from the enhanced Petri net and associated performance measures, we were unable to answer Q1 as PEP does not offer any support to profile the process evolution over time. We concluded that PEP is unable to answer Q1.

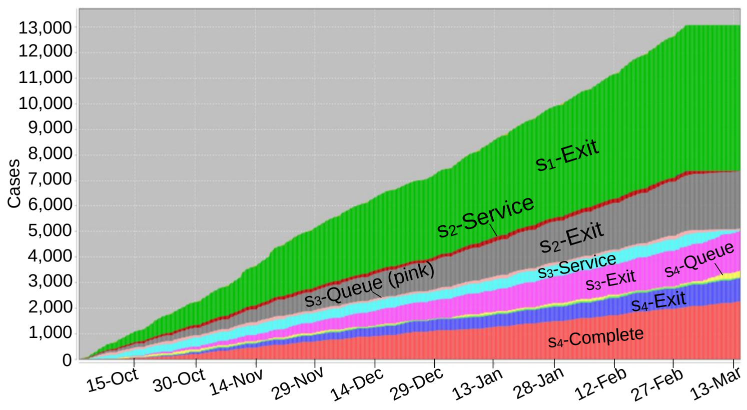


Fig. 8. CFD for the BPI Challenge 2012 log. Each stage s_i has a queue, service and exit flow. Some flows such as s_1-Queue, s_1-Service, s_2-Queue, and s_4-Service are fast and not observable on the normal scale.

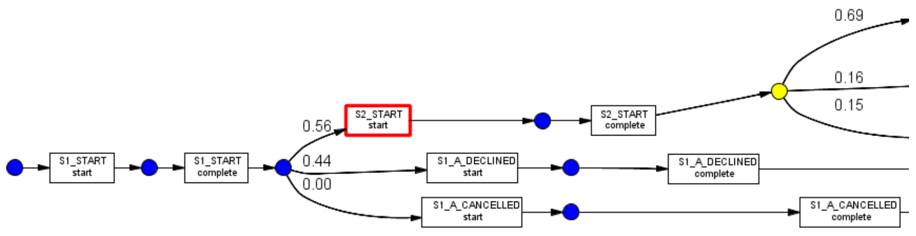


Fig. 9. Discovered Process Model in PEP, using gate events only.

Disco Similar to PEP, the model discovered by Disco from the unfiltered log was rather complex, with 50 activities and over 150 paths. Hence, we also decided to retain the gate events only, leading to a rather simple model with 11 activities and 19 paths (Fig. 10.a). Based on this model, we found two ways to answer Q1. One way was using the filter by timeframe provided by Disco to select different process variants by time intervals, e.g. by months from Oct 2011 to Mar 2012. After each interval, we recorded the performance measures manually for each process variant. At the end of this procedure, we obtained a set of monthly performance measures which we could use for trend analysis. While this approach could provide a precise measurement of process evolution, the results are not easy to retrieve and interpret from the figures manually calculated. We were unable to discover any insights because of the limitation of this manual review.

Another way was to animate the log on top of the discovered model, to identify any normal and abnormal trends (Fig. 10.b). While the animation was running, we had to keep close attention to the various tokens flowing towards different directions through the model, to identify recurring patterns as well as deviations. To complete the animation for six months, it took approximately four minutes at maximum speed which is a reasonable time. One insight was that the cases seem to flow to the end of the process in batches. However, it was not easy to pinpoint the recurrent timing of these batches during the animation. We were also unable to compute the volume of cases in batches due to the lack of supporting performance figures in the animation. In conclusion, we found that although the animation in Disco can provide some insightful clues w.r.t. to process evolution, it is not possible to precisely characterize this evolution.

Q2: How formation and dissolution of bottlenecks affects overall performance?

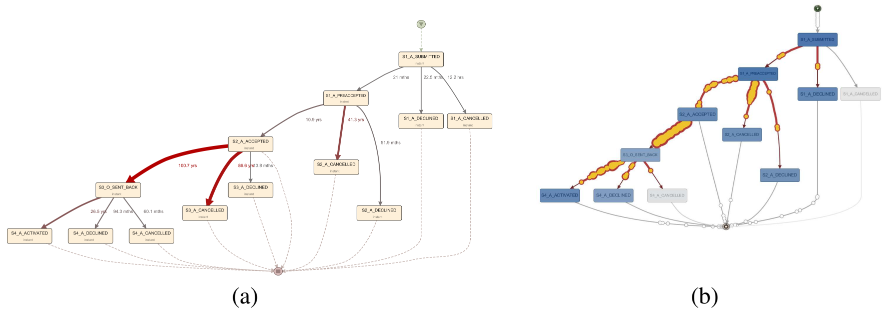


Fig. 10. (a) Filtered process model in Disco with highlighted bottlenecks and (b) its animation.

SPF We can observe signs of bottlenecks on the CFD when the queue band and/or service band become wider, meaning that the process has slower response to the arrival of new cases. The formation of bottlenecks can be identified from the time series charts of CIP and TIS of the queue and service stages, particularly at the peak points. The exact measurement of these effects is provided via the on-screen tooltips and by the PST with the time interval scale. The formation of bottlenecks generally leads to an increase of CIP and TIS in the queue and service period of a stage and possibly to a decrease of FE. Conversely, these effects gradually diminish when the bottleneck dissolves.

Although the log exhibits a stable process evolution, there are signs of bottlenecks. For example, Fig. 11.a shows that at stage s_4, the queue (s_4-Queue) widens from 24 Oct and peaks on 27 Oct (CIP=120 cases, see Fig. 12.b) and then slowly decreases onwards. The time series chart of the flow efficiency for this stage (see Fig. 12.a) also shows a fall on 26 and 27 Oct (around 0.55% fall as measured by the PST). Our measurement also shows that the CIP and TIS of s_4-Service do not increase immediately from 26-27 Oct (ca. CIP=27 cases, TIS=20 hours) but only afterwards (ca. CIP=42 cases, TIS=46 hours from 29 Oct to 6 Nov 2011) as the aftermath of the previous congestion (Fig.11.a). The bottleneck then slowly dissolves towards 16 Nov (Fig. 11.a) as the process increases its departure rate at s_4-Service after the bottleneck (from ca. 20 cases/day during 23-27 Oct to ca. 24 cases per day during 28 Oct-16 Nov). We observe that the FE has recovered and CIP and TIS have diminished during the period 28 Oct-16 Nov (Fig. 12.a and 12.b). Similar bottleneck phenomena are visible in stage s_4 at different times.

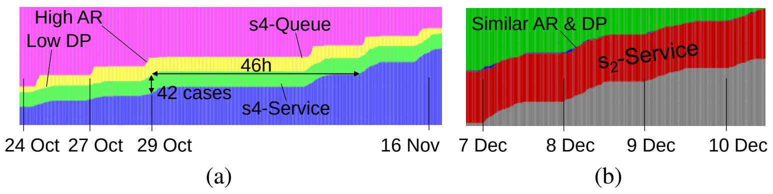
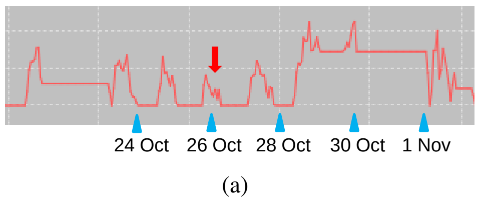


Fig. 11. (a) Example of widening queue at s_4 and (b) very minor queue at s_2.

In conclusion, with our approach it is easy to retrieve data with interpretable and precise information to answer Q2, deriving information on how bottleneck formation and dissolution affect process performance.

PEP Continuing from the enhanced model in Q1, PEP can highlight the bottlenecks on the model by coloring the places of the Petri net based on their



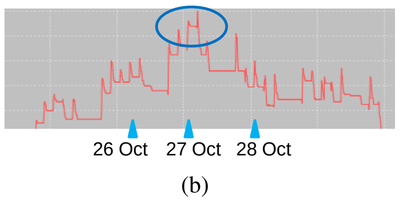
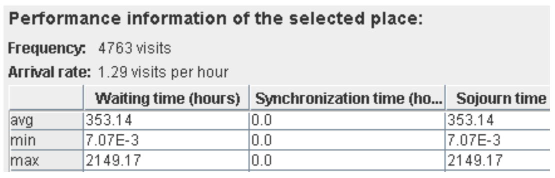


Fig. 12. (a) Flow Efficiency at s_4-Service and (b) CIP at s_4-Queue.

associated waiting time (see Fig. 9). This information is enriched by detailed performance measurements at the level of individual elements (see e.g. Fig.13).



Performance information of the selected place:

Frequency: 4763 visits

Arrival rate: 1.29 visits per hour

	Waiting time (hours)	Synchronization time (ho...	Sojourn time
avg	353.14	0.0	353.14
min	7.07E-3	0.0	7.07E-3
max	2149.17	0.0	2149.17

Fig. 13. Performance measures in PEP.

However, we found no ways to reason about the impact of the formation and dissolution of bottlenecks on the process performance as the measures shown on the model are only aggregate values over the whole log timespan. It is not possible to drill down to lower levels of granularity, e.g. checking the daily arrival rate at a given place, and profile this over time. Thus, we conclude that PEP is unable to answer Q2.

Disco Continuing from Q1 with the discovered high-level process model, we identified two ways of detecting bottlenecks in Disco. One is displaying performance measures on the model (Fig. 10.a). Disco can highlight in red the exceptionally high values of activity and path durations as signs of bottleneck. We found that the paths for canceled cases at stages s_2, s_3 and s_4 take too long, e.g. 21 days at stage s_3. In addition, the path for cases going from s_3 to s_4 is also longer than average (11.9 days). While the use of filters allow one to measure the impact of a bottleneck on overall performance (e.g. by measuring how much the average cycle time improves by removing slow cases), based on the process model and the performance measures alone, we did not have enough data to assess the impact of formation and dissolution of bottlenecks on overall performance.

Another way of answering Q2 is by watching the replay animation (Fig. 10.b). From this we can observe that there are busy flows of canceled cases at stages s_2, s_3 and s_4, and from s_3 to s_4. The tokens following these paths seem to be moving slower than those on other paths. However, we were unable to quantify these signs of bottleneck such as number of cases and waiting time, as well as the impact of these bottlenecks.

Q3: How do changes in demand and capacity affect overall process performance?

SPF The demand and capacity are represented by the arrival (AR) and departure rate (DR), respectively. The arrival rate at the first stage is the customer demand while the departure rates at different stages are their corresponding capacities. They can be observed on the CFD, as well as in the time series charts of these characteristics. Any change in these characteristics will affect the overall process performance, including the CIP, TIS and FE of the queue and service periods, and lead to the formation and dissolution of bottlenecks.

Overall, the PST in Fig. 6 shows that the process under exam has a much higher AR at s_1-Queue as customer demand rate (84.73 cases/day), than DR at s_4-Service as final output rate (18.62 cases/day). However, the process maintains a stable evolution without congestion because there are strong exit flows as shown in Fig. 8. This mechanism effectively reduces the strain of high customer demand on the process.

As such, the impact of demand and capacity is visible locally at a stage only. For example, in relation to the bottleneck reviewed in Q2, the differential chart in Fig. 14 shows that the bottleneck appears due to the stronger dominance of the arrival rate over the departure rate prior to the bottleneck period (14-24 Oct). The difference between arrival and departure patterns also has impact on the process performance. For example, Fig. 11.a shows that the arrival rate AR at s_4-Queue is high within a short time (ca. 14 cases/day) while the departure rate DR is low (ca. 5 cases/day) and spreads over a longer time. This difference explains why there is a permanent long queue before s_4-Service, which we identified when answering Q2. In contrast, the AR and DR at s_2-Queue are approximately equal with the same distribution (see Fig. 11.b). That is why there is a very minor queue at stage s_2.

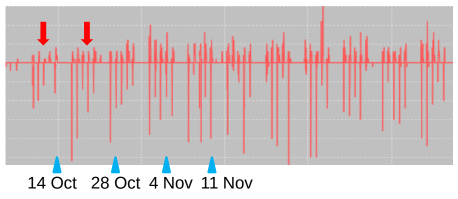


Fig. 14. Differential Chart (DR-AR) at s_4-Queue.

PEP We found no ways in PEP to investigate the impact of changes in demand and capacity on the process performance since this tool only captures one average value at every place/transition for the whole period. Hence, we are unable to answer Q3.

Disco We replayed the animation in Disco while focusing on the speed of the token flows at the start activity of each stage and tried to learn how this relates to the flow of tokens departing from the last activity of each stage (Fig. 10.b). However, we found it is very challenging to spot any patterns on the animation, since it is hard to capture the timing of tokens flowing at two different locations at the same time. We concluded that Disco is unable to answer Q3.

5 Conclusion

We presented an approach to analyze flow performance from event logs based on the concept of SPF, which transpose ideas found in lean management and agile software development to the field of PPM. The evaluation on real-life event logs puts into evidence qualitative advantages of this approach with respect to existing PPM techniques.

A key limitation of the SPF approach is the assumption that the log is divided into user-defined stages. In some cases, such stages may be already known (e.g. because they are captured in a process model), but in other scenarios the stages need to be discovered. A direction for future work is to design techniques for automated identification of *candidate stages* from a log. One possible approach is to cluster activities based on which resources most often perform them, as in [13]. An alternative is to cluster activities according to data dependencies, as in [17] where event types are grouped into clusters (corresponding to candidate sub-processes) based on shared data attributes.

Another limitation is that the approach still requires the user to manually identify patterns from the stage characteristics and visualizations, particularly patterns associated with formation and dissolution of bottlenecks. There is an opportunity to extend the SPF approach with techniques from statistical process control and change point analysis, such as CUSUM charts [18], to support the identification of such patterns. Another future work avenue is to conduct a usability evaluation of the SPF approach via controlled experiments in order to validate major design choices, such as the choice of stage characteristics and visualizations.

Acknowledgments This research is funded by the Australian Research Council Discovery Project DP150103356 and the Estonian Research Council (grant IUT20-55).

References

1. van der Aalst, W.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.: Fundamentals of Business Process Management. Springer (2013)
3. Modig, N., Ahlström, P.: This is lean: Resolving the efficiency paradox. Rheologica (2012)
4. Hornix, P.T.: Performance analysis of business processes through process mining. Master's thesis, Eindhoven University of Technology (2007)
5. Gunther, C.W., Rozinat, A.: Disco: Discover your processes. In: Proc. of BPM Demos. Volume 940 of CEUR Workshop Proceedings. (2012) 40–44
6. van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **2**(2) (2012) 182–192
7. van Dongen, B.F., Adriansyah, A.: Process mining: fuzzy clustering and performance visualization. In: Proc. of BPM Workshops, Springer (2010) 158–169
8. de Leoni, M., van der Aalst, W.P., Dees, M.: A general framework for correlating business process characteristics. In: Proc. of BPM, Springer (2014) 250–266
9. Pika, A., Wynn, M.T., Fidge, C.J., ter Hofstede, A.H., Leyer, M., van der Aalst, W.M.: An extensible framework for analysing resource behaviour using event logs. In: Proc. of CAiSE, Springer (2014) 564–579
10. Gunther, C.W., van der Aalst, W.M.P.: Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In: BPM 2007. 328–343
11. Conforti, R., Dumas, M., La Rosa, M., Maaradji, A., Nguyen, H., Ostovar, A., Raboczi, S.: Analysis of business process variants in apromore. In: Proceedings of the BPM Demos. Volume 1418., CEUR (2015)
12. Senderovich, A., Weidlich, M., Gal, A., Mandelbaum, A.: Queue mining - predicting delays in service processes. In: Proc. of CAiSE, Springer (2014) 42–57
13. de Smet, L.: Queue mining: Combining process mining and queueing analysis to understand bottlenecks, to predict delays, and to suggest process improvements. Master's thesis, Eindhoven University of Technology (2014)
14. Reinertsen, D.: Managing the Design Factory: A Product Developers Tool Kit. Simon & Schuster Ltd. (1998)
15. Nguyen, H., Dumas, M., ter Hofstede, A., La Rosa, M., Maggi, F.: Business process performance mining with staged process flows. "QUT ePrints Technical Report 91110" (`http://eprints.qut.edu.au/91110`), Queensland University of Technology (2015)
16. Venkatesh, V., Bala, H.: Technology acceptance model 3 and a research agenda on interventions. Decision Sciences **39**(2) (2008)
17. Conforti, R., Dumas, M., García-Bañuelos, L., La Rosa, M.: BPMN miner: Automated discovery of BPMN process models with hierarchical structure. Inf. Syst. **56** (2016)
18. Reynolds, M., Amin, R., Arnold, J.: CUSUM charts with variable sampling intervals. Technometrics **32**(4) (1990) 371–384