Stage-Based Process Performance Analysis

Chiao-Yun Li[1], Sebastiaan J. van Zelst[1,2], and Wil M.P. van der Aalst[1,2]

[1] Fraunhofer FIT, 53754 Sankt Augustin, Germany
{chiao-yun.li,sebastiaan.van.zelst,wil.van.der.aalst}@fit.fraunhofer.de
[2] RWTH Aachen University, Templergraben 55, 52062 Aachen, Germany
{s.j.v.zelst,wvdaalst}@pads.rwth-aachen.de

Abstract. Process performance mining utilizes the event data generated and stored during the execution of business processes. For the successful application of process performance mining, one needs reliable performance statistics based on an understandable representation of the process. However, techniques developed for the automated analysis of event data typically solely focus on one aspect of the aforementioned requirements, i.e., the techniques either focus on increasing the analysis interpretability or on computing and visualizing the performance metrics. As such, obtaining performance statistics at the higher level of abstraction for analysis remains an open challenge. Hence, using the notion of process stages, i.e., high-level process steps, we propose an approach that supports human analysts to analyze the performance at the process-stage-level. An extensive set of experiments shows that our approach, without much effort from users, supports such analysis with reliable results.

Keywords: Process performance analysis · Event abstraction · Process stage · Performance visualization.

1 Introduction

The goal of all the business is to maximize the return on investment, which can be realized through improving the efficiency of their business processes [10]. Analyzing process efficiency enables companies to locate and diagnose the causes of the bottlenecks in order to optimize workload scheduling and the distribution of resources. *Process mining* is a technology that empowers companies to analyze a process by exploiting an event log, i.e., records of events executed during the execution of a process [20]. *Process performance mining* is a subfield that focuses on the process performance, often referred to as the *time* dimension, to identify and diagnose the inefficiencies during the operation of a business process [4].

The effectiveness of the analysis of the process performance depends on two aspects, i.e., interpretability of the results as well as the reliability of the performance metrics provided. To achieve this, most process performance mining techniques project the performance information on a discovered or predefined process model [5,9,21]. However, as the behavior in a process becomes more

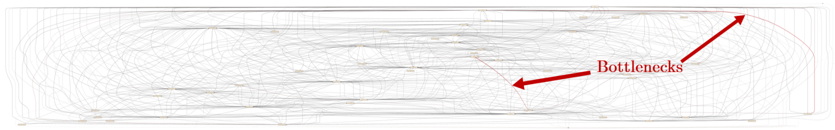


Fig. 1: Performance projection at activity level using Disco [5].

complex, the results may be no longer reliable and interpretable for human analysts. For example, as shown in Fig. 1, each of the bottlenecks highlighted only occurs once and the graph is difficult to read. As such, diagnosis remains difficult.

Problem Statement. Consider the medical domain, where specific activities are performed *prior*, *during*, and *after* the surgery. In some cases, the exact scheduling of activities within such a *part* of the process might be arbitrary, leading to the complex behavior. Such complexity results from the original level of granularity of the process, i.e., activities. As shown in Fig. 1, the complexity leads to the unreliable diagnosis due to the noninterpretable results and the relatively low frequency of the bottlenecks.

Moreover, an effective diagnosis of the inefficiencies in a process requires the context of cases. Consider the same example of performing a surgery in the medical domain. Assume that the average duration of conducting a surgery (*during*) is 5 hours and preparation for a surgery (*prior*) is 15 minutes. Suppose that the efficiency of conducting a surgery depends on how much the necessary materials or information are prepaid for the surgery in advance. Without the performance statistics presented in the context of a case, the bottleneck may be identified as performing the surgery. However, the cause of the bottleneck, i.e., how well the surgery is prepared in advance, may not be diagnosed. The paper aims to enable analysts to locate and diagnose the inefficiencies in complex processes by addressing the following questions:

Q1. What is the major *part* in a complex process that forms the bottleneck which *actually* causes the additional costs in the process?
Q2. What causes the bottleneck and how does the bottleneck impact the following *parts* of the process?
Q3. How reliable is the performance statistics given?

Solution Overview. In this paper, we formally define such *parts* as the concept of *stages* to elevate the analysis of process performance to the *stage* level. By only consider the activities that would determine the performance of a stage, we provide an overview of the stage performance with the throughput time of a stage and the time that the stage is *in active* in a process. The details of a stage, i.e., the executions within the stage, are not of concern considering such details may result in losing the focus of the big picture as shown in Fig. 1. Then, we visualize the performance metrics for the diagnosis at the stage level.

By doing so, the process can be easier understood and, thus, diagnosed at a higher level of abstraction. Moreover, the reliability of the results can be enhanced by including a sufficient number of performance measurements (compared with the bottleneck that occurs only once in the process). Meanwhile, to support the diagnosis, the visualizations emphasize on the performance statistics in the context of cases. To summarize, the proposed approach supports an analyst to identify the bottlenecks at the stage level and diagnose the root causes before drilling down the process at the original granularity level.

Our implementation and the datasets used for evaluation are available for replicating the experiments. We evaluate our solution by analyzing two event logs and compare the results with the existing techniques. The contributions are summarized as follows:

1. We develop an approach to extract the execution of stages defined by a user. According to our evaluation, the proposed approach reaches the balance between the usability and the reliability of the results compared with the existing techniques.
2. Provided the definition of stages, we introduce performance metrics that are straightforward for a user, allowing one to perform an unbiased analysis.
3. We provide a visualization that shows the evolution of the performance at the stage level and the interaction between stages. Meanwhile, the performance distribution of cases are presented together at a glance for diagnosis.

The remainder of the paper is organized as follows. Section 2 introduces related work in the field of process mining. The proposed approach is presented in Section 3. In Section 4, we perform analysis of stage performance by experimenting with various techniques and summarize the paper in Section 5.

2 Related Work

To the best of our knowledge, this is the first work that focuses on the performance analysis at a coarser granularity level by extracting the instances at the corresponding abstraction level and compute and visualize the duration accordingly. Using the existing techniques, such analysis may be performed by applying filtering or combining different techniques. Filtering based on per target coarse-granular instance may result in biased performance results. Alternatively, one may combine different techniques, which we classified into event abstraction and performance analysis techniques, to analyze the performance based on the instances extracted at a coarser granularity level.

Event Abstraction. Based on the output of the techniques, the event abstraction techniques can be further classified into model-based and non model-based. In [15], the authors decompose a process model into groups of *activities*, i.e., well-defined process steps, by mimicking the intuition of a human analyst identifying stages according to the modularity of the graph. Mannhardt and Tax identify the coarse-granular instances based on user-defined patterns which are represented by process models [14]. However, both approaches do not guarantee the availability of the results as shown in the experiments.

Other approaches do not require a process model. A supervised learning technique predicts the stage of an event using a probabilistic model [18,19]. The prediction model is trained with an event log in which all the events are labeled with the target instances at the coarser granularity level. Assuming that there exist patterns of the occurrence of the activities within a user-defined time interval, de Leoni and Dündar cluster activities and annotate the corresponding clusters to the events [12]. To analyze the performance of the process at the abstracted level, one has to apply other performance analysis techniques on top of the results of using the event abstraction techniques. For example, one may discover a process model at a higher granularity level from an abstracted event log and project the performance statistics on the model [11,9,21].

Performance Analysis. Similarly, we categorize the techniques into model-based and non model-based. Model-based techniques project the performance statistics on a predefined or discovered process model [5,9,21]. The performance can be analyzed with the context that is presented with the model. Nevertheless, due to the modeling formalism, i.e., certain process models do not allow for expressing all possible control-flow behaviors, the resulting process behavior highly depends on the discovery technique or the modeling method applied. The performance metrics may, thus, present biased results. Also, the aggregated performance metrics on a model limits the depth of the diagnosis. One needs to look into the cases in order to identify the root causes.

Other performance analysis techniques do not require a process model. The objectives of such methods vary and, therefore, it is hard to point out a common technique applied. Generally speaking, they aim to present unbiased results by showing the raw performance measures. For example, the *dotted chart* is a simple, yet powerful, technique that allows batched executions to be observed [17]. However, additional calculation is required to quantify the observed behavior and the context, i.e., relationship between activities, is lost. Another work focuses on the process performance over time with a parallel plot showing the duration of each process step on a absolute timeline [2,3]. The batched executions, which are often the causes of the delays in a process, can be easily observed and some behaviors, e.g., the overtake of the activities, may be discovered. The visualization emphasizes on the performance of and the interaction between process steps. However, without the context of a case, some diagnosis may be limited. For example, the influence of the performance of a process step to another one which is not directly following the one may not be observed and compared at the case level. The work is extended by incorporating a process model such that the performance of a process step can be analyzed with the context and more advanced process behavior may be presented with the model [1]. However, using a model suffers from the modeling formalism mentioned. In [16], the authors visualize the metrics such as the number of cases that arrive at each phase of a process per day. The workload and the efficiency in a phase over a specific time frame can then be observed. However, every event in an event log must be assigned to a phase and the phases must occur in a specific order, i.e., no phase could be skipped and no parallel phases is possible.

The existing event abstraction techniques may suffer from the biased results due to the modeling formalism or the assumptions of a process which the techniques are developed based on. The current performance analysis techniques are either insufficient for analyzing the influence of bottlenecks in the context of cases, one of the objectives of our research, or restricted to the assumed process behavior. To conclude, simply combining the existing techniques is insufficient to analyze the process performance at a coarser granularity level.

3 Stage-Based Performance Mining

An overview of the approach is presented in Fig. 2. It is a two-fold approach, which extracts stage instances, i.e., the execution of stage classes, and visualizes the performance metrics. The approach consists of four core components: *Mine for Stage Instances*, *Compute Stage Performance Metrics Visualize Stage Performance Evolution* and *Visualize Stage Performance Summary*. Based on the stage classes specified by a user, *Mine for Stage Instances* extracts the stage instances. The performance metrics is computed and visualized with *Visualize Stage Performance Evolution* and *Visualize Stage Performance Summary*. This section formally defines the terms mentioned and explains the components shown in Fig. 2 after briefly introducing the basic concept used in our approach.

3.1 Preliminaries

Given an arbitrary set X, we write $\mathscr{P}(X)=\{X'|X'\subseteq X\}$ to denote its powerset. A sequence of length n over X is a function $\sigma:\{1,2,...n\}\rightarrow X$. Let X^* denote the set of all sequences over X. We write $\sigma=\langle x_1,x_2,...,x_n\rangle\in X^*$, where $\sigma(1)=x_1,\sigma(2)=x_2,...,\sigma(n)=x_n$. Given a sequence $\sigma\in X^*$, $|\sigma|$denotes the length of the sequence. The empty sequence is written as $\langle\rangle$, i.e., $|\langle\rangle|=0$. We overload the set notation and write $x\in\sigma$ if and only if $\exists 1\leq i\leq|\sigma|\big(\sigma(i)=x\big)$.

In a process, an execution of an *activity* is recorded as an *event* with the timestamp of the execution in the context of a process instance, i.e., a *case*. The events of a process are collected in an *event log*, the input for any process mining technique. In practice, many additional data attributes can be associated with an event. For example, event data typically captures the resource executing the activity, the cost of such an activity, etc. In this paper, we represent an event by a pair $e=(a,t)$ executed in the context of a case represented by a trace. The definitions of an event, trace, and event logs are as follows.

Definition 1 (Event, Trace & Event Log). *Let $\mathscr{A}$ denote the universe of process activities and $\mathscr{T}$ denote the universe of time. An event $e=(a,t)\in\mathscr{A}\times\mathscr{T}$ represents the execution of activity a at time t. We let $\mathscr{E}=\mathscr{A}\times\mathscr{T}$ denote the*

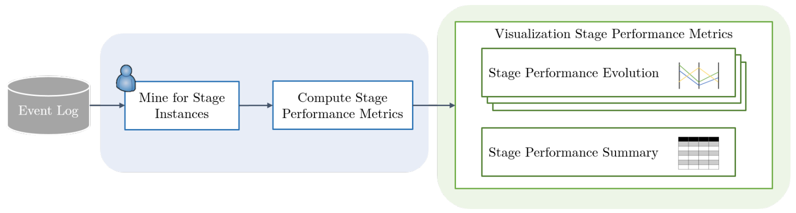

Event Log
Mine for Stage Instances
Compute Stage Performance Metrics
Visualization Stage Performance Metrics
Stage Performance Evolution
Stage Performance Summary

Fig. 2: Schematic overview of the proposed approach.

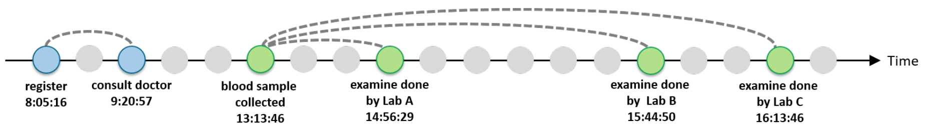


Fig. 3: A trace of a medical process of a patient in a hospital.

universe of events. Given $e{=}(a,t){\in}\mathscr{E}$, we let $\pi_{act}(e){=}a$ and $\pi_{ts}(e){=}t$. A trace σ *is a sequence of events, i.e., $\sigma{\in}\mathscr{E}^*$, such that $\forall 1{\le}i{<}j{\le}|\sigma|\ (\pi_{ts}(\sigma(i)){\le}\pi_{ts}(\sigma(j)))$. An* event log *$L$ is a collection of traces, i.e., $L{\subseteq}\mathscr{E}^*$.*[3]

3.2 Mine for Stage Instances

A natural way to analyze the performance of a complex process is to firstly elevate the process to the *stage* level. The performance of such a stage might not be impacted (significantly) by the arbitrary scheduling of activities inside a stage. Therefore, only the activities that a stage might start and end with are of interest when analyzing the performance at a higher level of abstraction, i.e., *at the stage level.*

Meanwhile, the level of abstraction of a process depends on the organization and the analysis objectives. For example, for a process operated by several companies, a stage can be defined as the part of the process within a specific company or a department of a company. This depends on the question of interest. To support such an analysis, the proposed approach allows analysts to specify the stages with the activities that a *stage class* may start and end with. The execution of stage classes, i.e., *stage instances*, are the actual operations in a case. We formally define a stage class and a stage instance below.

Definition 2 (Stage Class). *A stage class S is a pair of non-empty sets of disjoint activities, i.e., $S{=}(A_s, A_c){\in}(\mathscr{P}(\mathscr{A})\setminus\{\emptyset\}){\times}(\mathscr{P}(\mathscr{A})\setminus\{\emptyset\}){\wedge}A_s{\cap}A_c{=}\emptyset$, where A_s (start activities) represents the activities that the stage class may start with and A_c (end activities) represents the activities that the stage class may end with. We let $\mathcal{S}{=}(\mathscr{P}(\mathscr{A})\setminus\{\emptyset\}){\times}(\mathscr{P}(\mathscr{A})\setminus\{\emptyset\})$ denote the set of all stage classes.*

Definition 3 (Stage Instance). *Let $\sigma{\in}\mathscr{E}^*$ be a trace and $S{=}(A_s, A_c){\in}\mathcal{S}$ a stage class. We define a function $\gamma{:}\mathscr{E}^*{\times}\mathcal{S}{\to}\mathscr{P}(\mathscr{E}{\times}\mathscr{E})$ that returns a set of pairs of events such that $\gamma(\sigma, S){\subseteq}\{(\sigma(i), \sigma(j))|1{\le}i{<}j{\le}|\sigma|{\wedge}\pi_{act}(\sigma(i)){\in}A_s{\wedge}\pi_{act}(\sigma(j)){\in}A_c\}$. Each pair of events $si{=}(e, e'){\in}\gamma(\sigma, S)$ is a stage instance. For simplicity, we write $\gamma(\sigma, S)$ as $\gamma_\sigma(S)$.*

How to extract the stage instances depends on the business context of the analysis. Consider an example of a medical process. Suppose that we are interested in the duration of a patient being registered in the hospital until the doctor consultation and the time for the laboratories to examine the blood sampled from the patient. We define two stage classes, $S_1{=}(\{\textit{registered}\}, \{\textit{consult doctor}\})$ and

[3]We assume that an event can only appear once in a trace and that no two cases have the same trace in an event log. This can be enforced by adding more event attributes.

S_2=({*sample collected*}, {*examine done by Lab* $*$}), where $*$ denotes any string fitting the pattern. Fig. 3 illustrate the trace of a case, where the dots denote the events in the trace. The events of interest are labeled and colored in blue and green for two stage classes. The grey dots are the activities, e.g., consulting nurses, that are not of concern for the analysis. The stage instances are the pairs of events connected with the dashed lines. The example shows that there are many possible ways to extract stage instances, depending on the process and the objectives of analysis.

The realization of extracting stage instances is by applying a generic technique that we developed in [13]. We extract the maximal number of stage instances in a trace. Considering the scenario mentioned, we allow one to flexibly define how the events of the start and the end activities should be mapped. We assume that the closer two events are, the more likely they form a stage instance. Such *distance* between events can be specified based on domain knowledge. Without the knowledge of how the events should be paired and the distance specified, we assume one-to-one mapping of events and use the order of the events in a trace by assuming that the closer two events are in a trace, the more possible that they belong to the same stage instance.

3.3 Compute Stage Performance Metrics

Given the stage instances extracted, the duration of a stage instance i, i.e., *cycle time* is naturally derived, which indicates the duration that a stage class is *in active*. To differentiate the duration that a stage class is actually being executed and the idle time of a stage class, we define *flow time* as the first occurrence of *any* start activity of the stage class until the last occurrence of *any* end activity of the stage class. Moreover, to quantify the behavior among stage classes executed, we introduce the metrics for inter-stage performance. The metrics is formally defined as follows.

Definition 4 (Cycle Time). *Given an event log L and a stage class S, cycle time (ct) is the duration of a stage instance $si \in \gamma_\sigma(S)$ where $\sigma \in L$, i.e., $ct_S(si) = \pi_{ts}(si(2)) - \pi_{ts}(si(1))$. For each trace, we aggregate all the $ct_S(si)$, $\forall si \in \gamma_\sigma(S)$, into $ct_S^{stat}(\sigma)$, where stat stands for the target statistics, e.g., $ct_S^{avg}(\sigma)$ refers to the average duration of $ct_S(si)$, $\forall si \in \gamma_S(\sigma)$. For the process, we collect all the cycle time of S in L, i.e., $CT_S(L) = [ct_S(si) | \forall si \in \gamma_\sigma(S), \sigma \in L]$, and aggregate into $CT_S^{stat}(L)$.*

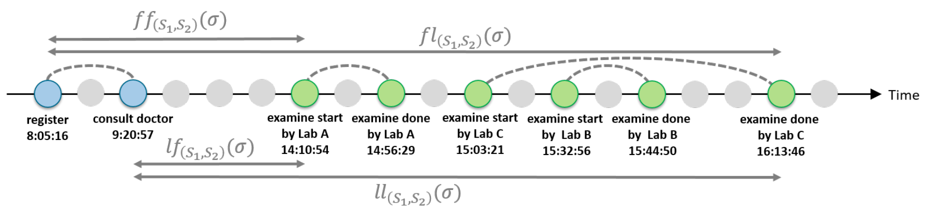

$ff_{(S_1,S_2)}(\sigma)$
$fl_{(S_1,S_2)}(\sigma)$
Time
register 8:05:16
consult doctor 9:20:57
examine start by Lab A 14:10:54
examine done by Lab A 14:56:29
examine start by Lab C 15:03:21
examine start by Lab B 15:32:56
examine done by Lab B 15:44:50
examine done by Lab C 16:13:46
$lf_{(S_1,S_2)}(\sigma)$
$ll_{(S_1,S_2)}(\sigma)$

Fig. 4: A trace of a medical process of a patient in a hospital.

Table 1: Implication of behavior between two stage classes S_1 and S_2 in a trace σ based on the inter-stage performance metrics.

Metrics Relation	Implication of Stage Behavior in a Trace σ
$lf_{(S_1,S_2)}(\sigma) > 0$	S_2 starts after S_1 terminates permanently.
$fl_{(S_1,S_2)}(\sigma) > 0$	S_2 terminates permanently before S_1 starts.
$lf_{(S_1,S_2)}(\sigma) < 0 \wedge ff_{(S_1,S_2)}(\sigma) > 0$	S_2 *may* be executed in parallel with S_1.

Definition 5 (Flow Time). *Given an event log L of a process and a stage class $S{=}(A_s, A_c)$, flow time (ft) is the duration that a stage class lasts in a case, i.e., $\forall 1{\le}i{<}j{\le}|\sigma|$ where $\sigma{\in}L$, $ft_S(\sigma){=}\pi_{ts}(\sigma(max(\{j|\pi_{act}(\sigma(j)){\in}A_c\}))){-}\pi_{ts}(\sigma(min(\{i|\pi_{act}(\sigma(i)){\in}A_s\})))$ if and only $\gamma_\sigma(S){\neq}\emptyset$. In other words, flow time only exists when S is executed, i.e., $\gamma_\sigma(S){\neq}\emptyset$. For the process, we aggregate all the flow time of S in L into $FT_S^{stat}(L)$, where stat stands for the target statistics.*

Consider a process of applying for a mortgage. An application, i.e., a case with its trace σ, is finally approved after several rejections and re-submissions (S). The flow time $ft_S(\sigma)$ represents the duration from the first submission until it is finally approved and the duration that the application is under review is $ct_S^{sum}(\sigma)$.

Definition 6 (Inter-Stage Performance Metrics). *Given a trace σ and a stage class $S{=}(A_s, A_c)$ which $\gamma_\sigma(S){\neq}\emptyset$, $\forall 1{\le}i{<}j{\le}|\sigma|$, we obtain two timestamps $t_S^{min}(\sigma){=}\pi_{ts}(\sigma(min(\{i|\pi_{act}(\sigma(i)){\in}A_s\})))$ and $t_S^{max}(\sigma){=}\pi_{ts}(\sigma(max(\{j|\pi_{act}(\sigma(j)){\in}A_c\})))$. Let S_1 and S_2 be two stage classes which $\gamma_(\sigma)(S_1){\neq}\emptyset{\wedge}\gamma_(\sigma)(S_2){\neq}\emptyset$. We compute the performance between S_1 and S_2 with the following metrics:*

- $ff_{(S_1,S_2)}(\sigma) = t_{S_2}^{min}(\sigma) - t_{S_1}^{min}(\sigma)$
- $fl_{(S_1,S_2)}(\sigma) = t_{S_2}^{max}(\sigma) - t_{S_1}^{min}(\sigma)$
- $lf_{(S_1,S_2)}(\sigma) = t_{S_2}^{min}(\sigma) - t_{S_1}^{max}(\sigma)$
- $ll_{(S_1,S_2)}(\sigma) = t_{S_2}^{max}(\sigma) - t_{S_1}^{max}(\sigma)$

Given the similar scenario in a hospital with a trace σ of a case, we define S_1=({*registered*}, {*consult doctor*}) and S_2=({*examine start by Lab* $*$, *examine done by Lab* $*$}. Fig. 4 visualizes the four metrics, $ff_{(S_1,S_2)}(\sigma)$, $fl_{(S_1,S_2)}(\sigma)$, $lf_{(S_1,S_2)}(\sigma)$, and $ll_{(S_1,S_2)}(\sigma)$. Since $lf_{(S_1,S_2)}(\sigma)$ is greater than zero , we know that, after the consultation, the first examination of the blood sample starts roughly 5 hours later. Table 1 summarizes the implication of the behavior between two stage classes based on the inter-stage performance metrics.

3.4 Visualize Stage Performance Metrics

We introduce two visualizations, *stage performance evolution* and *stage performance summary*. The first one demonstrates the evolution of performance over stages executed, allowing for further diagnosis. The latter one summarizes the statistics of the performance of all the stage classes defined. This section introduces the visualizations and demonstrates how the analysis can be performed with the visualizations using an event log L and four stage classes $\mathcal{S}$={*Apply*, *Claim*, *Travel*, *Declare*}.

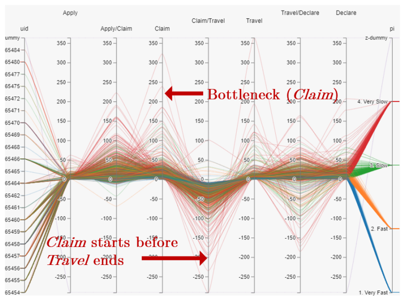
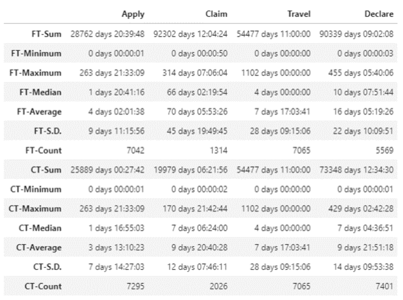


Fig. 5: Stage performance evolution.



	Apply	Claim	Travel	Declare
FT-Sum	28762 days 20:39:48	92302 days 12:04:24	54477 days 11:00:00	90339 days 09:02:08
FT-Minimum	0 days 00:00:01	0 days 00:00:50	0 days 00:00:00	0 days 00:00:03
FT-Maximum	263 days 21:33:09	314 days 07:06:04	1102 days 00:00:00	455 days 05:40:06
FT-Median	1 days 20:41:16	66 days 02:19:54	4 days 00:00:00	10 days 07:51:44
FT-Average	4 days 02:01:38	70 days 05:53:26	7 days 17:03:41	16 days 05:19:26
FT-S.D.	9 days 11:15:56	45 days 19:49:45	28 days 09:15:06	22 days 10:09:51
FT-Count	7042	1314	7065	5569
CT-Sum	25889 days 00:27:42	19979 days 06:21:56	54477 days 11:00:00	73348 days 12:34:30
CT-Minimum	0 days 00:00:01	0 days 00:00:02	0 days 00:00:00	0 days 00:00:01
CT-Maximum	263 days 21:33:09	170 days 21:42:44	1102 days 00:00:00	429 days 02:42:28
CT-Median	1 days 16:55:03	7 days 06:24:00	4 days 00:00:00	7 days 04:36:51
CT-Average	3 days 13:10:23	9 days 20:40:28	7 days 17:03:41	9 days 21:51:18
CT-S.D.	7 days 14:27:03	12 days 07:46:11	28 days 09:15:06	14 days 09:53:38
CT-Count	7295	2026	7065	7401

Fig. 6: Stage performance summary.

Stage Performance Evolution. It may occur that some cases execute some stages while others do not. We consider that the execution of stage classes reflects the business context. It is not reasonable to compare the performance of the cases without identifying different scenarios. Thus, we visualize the performance based on different *types* of cases according to the stage classes executed.

For each combination of stage classes executed, we visualize the stage performance metrics for the cases executing all the stage classes in the combination using parallel coordinates as shown in Fig. 5 [7]. The leftmost coordinate is the organization handling the cases and the rightmost one is the total case throughput time classified into *Very Slow*, *Slow*, *Fast*, and *Very Fast*. Between the two coordinates, the performance metrics of each trace $\sigma \in \{\sigma | \forall \sigma \in L \forall S \in \mathcal{S}, \gamma_\sigma(S) \neq \emptyset\}$ is plotted with a horizontal folded line in the order of $ct^{sum}_{Apply}(\sigma)$, $lf_{(Apply,Claim)}(\sigma)$, $ct^{sum}_{Claim}(\sigma)$, $lf_{(Claim,Travel)}(\sigma)$, $ct^{sum}_{Travel}(\sigma)$, $lf_{(Travel,Declare)}(\sigma)$, $ct^{sum}_{Declare}(\sigma)$ in the figure. The visualization can be applied interactively as below:

- The order of the coordinates can be flexibly arranged and the metrics of every stage class or between two stage classes can be changed to the flow time or other metrics, which allows for exploring the behavior of the stage performance from different angles.
- Depending on the use cases, the leftmost and rightmost coordinates may be replaced with any case attributes for analyzing the relationships between the attributes, e.g., the financial costs of handling a case, and the stage performance evolution.
- The scale of the coordinates for the performance metrics can be set the same (*absolute*) for identifying the bottlenecks, or the maximum value for each metrics (*relative*) such that the cause of the bottlenecks may be diagnosed.

Fig. 5 shows the visualizations using relative performance with the analysis. Suppose one assumes that the stages are executed one after another, i.e., no parallelism of stages. In Fig. 5, the bottleneck and the most severe deviation are identified based on the stage performance distribution of the cases.

Stage Performance Summary. To have an overview of the performance of all the stage classes defined, we summarize the performance for all the cases in L. As shown in Fig. 6, the summary is presented with the statistics of $FT^{stat}_S(L)$ and $CT^{stat}_S(L)$ for every stage class $S \in \mathcal{S}$.

4 Evaluation

With the aim of supporting analysts to identify the bottlenecks and perform the diagnosis of a complex process, we conduct a comparative evaluation based on two criteria: the ease of use of a method and the reliability of the metrics. A method that requires much preparation, manipulation of an event log, or the domain knowledge hampers an analyst to perform an effective analysis. The metrics that contains only a few measurements may cause misleading conclusion of the bottlnecks. In the evaluation, we conduct experiments by applying various techniques to perform analysis at a coarser granular level of a process. The techniques are compared with the following questions:

– How much is the manipulation of an event log required to analyze the process performance at a coarser granular level?
– How much domain knowledge is required for the abstraction of an event log?
– How reliable is the resulting performance metrics?

To the best of our knowledge, the stage performance evolution proposed is the only visualization that supports our goal for such analysis. Therefore, we conduct the experiments by using various event abstraction techniques of which our visualization is applied on top. Table 2 lists the implementations of the techniques evaluated with their abbreviations for convenience. Except for the proposed approach[4], other techniques are available in ProM [24]. This section presents the evaluation from the aspects of the ease of use and the reliability by analyzing two event logs, *PermitLog* [23] and *BPIC15_1* [22].

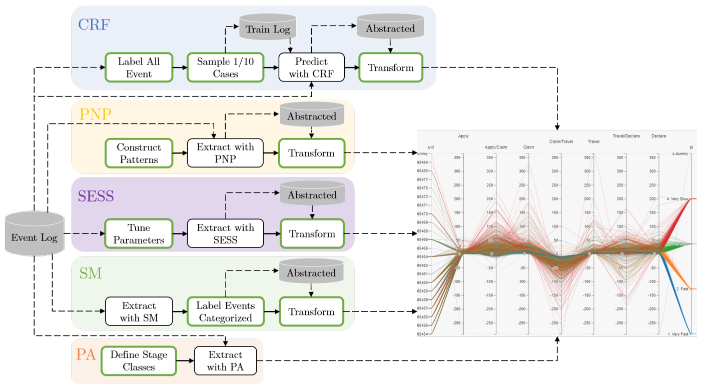


Fig. 7: A schematic overview of analyzing performance at a coarser granular level using different event abstraction techniques.

[4]The implementation and the datasets used for experiments are in `https://github.com/chiaoyunli/spm`.

Table 2: Overview: Techniques used for Experiments.

Techniques	Abbreviation
- Proposed Approach	PA
- Abstract Event Labels using linear chaining [18,19] Conditional Random Field (GRMM)	CRF
- Log Abstraction - Abstract Log based on Patterns [16]	PNP
- Session-based Log Abstraction [12]	SESS
- Stage Mining (SM) [15]	SM

4.1 Evaluation on Ease of Use

The ease of use of a tool is evaluated from two aspects, the amount of the domain knowledge required and the necessity of the manipulation of an event log. The inputs and outputs of the event abstraction techniques vary. Therefore, for each technique, we manipulate the event logs for performance analysis at the abstracted level if necessary.

External Effort Required. Fig. 7 presents the overview of the steps to analyze the performance at the coarser granularity level using the techniques. The dashed line indicates the data flow and the solid line refers to the control flow. Each box represents a step and the steps that require human intervention are emphasized with the green outline of the steps. We further group the steps and annotate the groups with the corresponding techniques. Since the existing event abstraction techniques are not specifically designed for performance analysis, we manipulate the output of the event abstraction techniques to compute the performance metrics (*Transform* step). If the output does not contain the attribute to indicate the instance of a concept at a coarser level of a process, we consider the continuous events with the same targeting instance at a coarser granularity level as an instance, i.e., the duration between the first and the last event of such instance corresponds to the *cycle time* in our approach. For other metrics, we apply the same definition in our approach, e.g., flow time of an instance of a higher level concept is the duration from the first to the last event of which the activities are contained in the high-level concept identified. As shown in Fig. 7, the proposed approach, i.e., PA, requires the least steps and does not require any transformation for the performance visualization. Note that other performance analysis techniques can be applied to analyze other aspects of the performance. In this case, our approach can, alternatively, generate an event log consisting of the events in the stage instances and the *Transform* step should be applied like the other techniques.

Domain Knowledge Required. The domain knowledge required for each technique varies. For example, to train a prediction model, CRF requires an event log with every event being labeled; for PNP, a coarse-granular instance is extracted with a pattern of the activities. To compare the domain knowledge quantitatively, we calculate the percentage of the activities required to extract a coarse-granular instance. Table 3 shows how the domain knowledge is required for every technique evaluated and the corresponding number of activities in the

Table 3: Overview: Techniques used for Experiments.

	Domain Knowledge Required	#Activities Required (%) (*PermitLog*/*BPIC15_1*)
CRF	All events labeled with the coarse-granular instance for training	1/1
PNP	Behavior of the activities of every concept at a coarser granular level	0.71/0.9
PA	Start and end activities of a stage class	0.37/0.78
SESS	Parameters tuning	0/0
SM	Minimum number of activities in a concept at a coarser granular level	0/0

input of the techniques in the experiments. Our approach outperforms the CRF and PNP. However, it is inferior to SESS and SM since the two techniques are unsupervised. Nevertheless, SESS requires exhaustively tuning of the parameters and the results are non deterministic. SM, as presented in the next section, cannot guarantee the availability of the results.

4.2 Evaluation on Metrics Reliability

We perform analysis using the methods based on the steps illustrated in Fig. 7. To evaluate the results, generally speaking, the accuracy is an ideal indicator of the reliability of the results. However, due to the assumptions of different techniques, e.g., some are supervised while others are unsupervised approaches, it is unfair to compare the accuracy for the reliability. Therefore, the experiments are conducted on a best effort basis and we consider the number of measurements included as the indicator for the reliability of the metrics, i.e., the more measurements and cases used to compute a performance metrics, the more reliable the results are. Note that, except for SM of which the results are unavailable, all the techniques require a user to determine the number of concepts in a coarser granular level, i.e., the number of stage classes in terms of the proposed approach. Therefore, for the concepts at a coarser granular level, we define four concepts for a travel reimbursement management process for *PermitLog* [8] and nine phases which are implied in activity code in a dutch municipality for *BPIC15_1* [6]. The quality of the results are examined from two perspectives, whether the number of the concepts identified matches with the number of the concepts defined and the amount of the measurements.

Table 4 presents the performance statistics with the number of the measurements for the cycle time and the cases executing the concepts at a coarser granular level, i.e., $\#ft$. For both event logs, CRF and SESS cannot extract the exact number of concepts defined. CRF identifies too many concepts which include the events that the technique fails to predict (None) using *PermitLog* and too less concepts using *BPIC15_1*. SESS extracts less clusters despite the fact that the numbers of clusters desired are specified with the parameter. Therefore, only PNP generates the same number of concepts at a coarser granular level as specified. However, only the results using *PermitLog* are available while they are inferior to the proposed approach in terms of the number of measurements included. To conclude, the proposed approach provides the most reliable metrics compared with the other techniques in the experiments.

Table 4: Number of measurements per high-level concept identified using *PermitLog* and *BPIC15_1*. NaN indicates that the results are unavailable.

(a) *PermitLog*

High-Level Concept Identified (#ct/#ft)				
CRF	PNP	PA	SESS	SM
- Apply (7911/7062)	- Apply (7911/7062)	- Apply (7911/7062)	- Start trip+ (5406/3965)	NaN
- Claim (1715/1336)	- Claim (1605/1296)	- Claim (2026/1314)	- Permit FINAL_APPROVED by SUPERVISOR+ (5715/4095)	
- Travel (7843/7065)	- Travel (6331/633)	- Travel (7065/7065)		
- Declare (5980/5718)	- Declare (5043/4963)	- Declare (7401/5569)	- Request Payment+ (10512/5856)	
- None (1276/1276)				

(b) *BPIC15_1*

High-Level Concept Identified (#ct/#ft)				
CRF	PNP	PA	SESS	SM
- Phase 1 (29/29)	NaN	- Phase 0 (1992/1199)	- register submission date request+complete (901/670)	NaN
- Phase 2 (29/29)		- Phase 1 (3967/1119)	- enter senddate decision environmental permit+complete (1498/948)	
- Phase 3 (193/178)		- Phase 2 (2727/969)	- registration date publication+complete (105/102)	
- Phase 4 (200/178)		- Phase 3 (2573/1028)	- enter senddate procedure confirmation+complete (100/97)	
- Phase 5 (180/176)		- Phase 4 (3397/925)	- enter senddate acknowledgement+complete (106/102)	
- Phase 8 (1027/1027)		- Phase 5 (2054/899)	- generate publication document decision environmental permit+complete (154/147)	
		- Phase 6 (1/1)	- create subcases completeness+complete (18/18)	
		- Phase 7 (138/138)		
		- Phase 8 (156/153)		

4.3 Experiments Summary

We perform a comparative evaluation by analyzing stage performance using various techniques. We compare the ease of the use of the techniques and the reliability of the resulting performance metrics. In terms of the ease of use, our approach requires the least effort from a user. However, we still require some domain knowledge in comparison with the unsupervised techniques. The reliability of the metrics is based on whether the number of the concepts at a coarser granular level is the same as specified and the number of measurements. The proposed approach outperforms all the other techniques evaluated. To conclude, the results show that our approach meets the balance between the ease of use and the reliability of the metrics.

4.4 Threats to Validity

The existing techniques are not designed for analyzing the performance at a high level of a process. Therefore, some information that is required to compute the duration of a coarse-granular instance, i.e., the start and complete time of the instance, is left for users to determine. Consider two interleaving instances of two concepts at a coarser granular level. Such behavior may result in multiple cycle time for each instance in the *Transform* step. However, in fact, only two measurements should be extracted. Thus, despite the best effort to apply the techniques, the results may not be accurate due to the manipulation.

For the proposed approach, the implementation allows an analyst to define only the stage classes with the distance and the mapping of events configured as default. However, there may be some scenarios where the parameters may not be defined easily and, thus, require further effort to configure the parameters to obtain reliable results. In addition, the performance of stage instances is aggregated at the case level. Which metrics makes sense for the analysis depends on the context. For example, in terms of stage instances of a stage class executed in parallel, the average cycle time may not be a reasonable choice for some processes. Nevertheless, consider the scenario in Fig. 3, the average can be used to compute the costs for hiring the staff in the laboratories. Such decision requires analysts to be aware of the context.

5 Conclusion

The diagnosis of inefficiencies requires performance metrics provided based on interpretable results. We elevate the analysis to the stage level and visualize the performance accordingly. Existing techniques are insufficient for stage performance analysis. The evaluation shows that combining existing techniques requires additional manipulation of an event log and domain knowledge from a user. Moreover, the results may be unreliable or unavailable. We propose an approach that supports performance analysis at the stage level by extracting events that are critical for the metrics. As such, our approach minimizes the effort from users while providing the most reliable results compared to the existing works. Meanwhile, the technique can be flexibly combined with other visualization to analyze other aspects of a process. To facilitate the analysis at the stage level, further research aims at automatic identification of stage classes.

References

1. van der Aalst, W., Unterberg, D.T.G., Denisov, V., Fahland, D.: Visualizing token flows using interactive performance spectra. In: International Conference on Applications and Theory of Petri Nets and Concurrency (2020)
2. Denisov, V., Belkina, E., Fahland, D., van der Aalst, W.: The performance spectrum miner: Visual analytics for fine-grained performance analysis of processes. In: BPM (Dissertation/Demos/Industry) (2018)
3. Denisov, V., Fahland, D., van der Aalst, W.: Unbiased, fine-grained description of processes performance from event data. In: International Conference on Business Process Management (2018)
4. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management (2018)
5. Günther, C.W., Rozinat, A.: Disco: Discover your processes. BPM (Demos) (2012)
6. van der Ham, U.: Benchmarking of five dutch municipalities with process mining techniques reveals opportunities for improvement (2015)
7. Haziza, D., Rapin, J., Synnaeve, G.: Hiplot, interactive high-dimensionality plots. `https://github.com/facebookresearch/hiplot` (2020)

8. Hobeck, R., Pufahl, L., Binetruy, P., Chada, W., Digtiar, M., Gülle, K.J., Slarzynska, M., Stiehle, F., Weber, I.: Performance, variant, and conformance analysis of an academic travel reimbursement process (2020)
9. Hornix, P.T.: Performance analysis of business processes through process mining. Master's Thesis, Eindhoven University of Technology (2007)
10. Kasim, T., Haracic, M., Haracic, M.: The improvement of business efficiency through business process management. Economic Review: Journal of Economics and Business (2018)
11. Leemans, S.J., Fahland, D., van der Aalst, W.: Discovering block-structured process models from event logs containing infrequent behaviour. In: International conference on business process management (2013)
12. de Leoni, M., Dündar, S.: Event-log abstraction using batch session identification and clustering. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing (2020)
13. Li, C.Y., van Zelst, S.J., van der Aalst, W.: A generic approach for process performance analysis using bipartite graph matching. In: International Conference on Business Process Management (2019)
14. Mannhardt, F., Tax, N.: Unsupervised event abstraction using pattern abstraction and local process models. arXiv preprint arXiv:1704.03520 (2017)
15. Nguyen, H., Dumas, M., ter Hofstede, A.H., La Rosa, M., Maggi, F.M.: Stage-based discovery of business process models from event logs. Information Systems (2019)
16. Nguyen, H., Dumas, M., ter Hofstede, A., La Rosa, M., Maggi, F.: Business process performance mining with staged process flows. In: International Conference on Advanced Information Systems Engineering (2016)
17. Song, M., van der Aalst, W.: Supporting process mining by showing events at a glance. In: Proceedings of the 17th Annual Workshop on Information Technologies and Systems (2007)
18. Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.: Mining process model descriptions of daily life through event abstraction. In: Proceedings of SAI Intelligent Systems Conference (2016)
19. Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.: Event abstraction for process mining using supervised learning techniques. In: Proceedings of SAI Intelligent Systems Conference (2016)
20. van der Aalst, W.: Process mining: Data science in action. Springer (2016)
21. 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 (2012)
22. van Dongen, B.: BPI challenge 2015 municipality 1 (2015). https://doi.org/"10.4121/uuid:a0addfda-2044-4541-a450-fdcc9fe16d17"
23. van Dongen, B.: BPI challenge 2020: Travel permit data (2020). https://doi.org/10.4121/uuid:ea03d361-a7cd-4f5e-83d8-5fbdf0362550
24. van Dongen, B., de Medeiros, A.K.A., Verbeek, H., Weijters, A., van der Aalst, W.: The prom framework: A new era in process mining tool support. In: International conference on application and theory of petri nets (2005)