Temporal Performance Analysis for Block-Structured Process Models in Cortado

Chapter · May 2022

DOI: 10.1007/978-3-031-07481-3_13

4 authors, including:

Daniel Schuster
Fraunhofer Institute for Applied Information Technology FIT
16 PUBLICATIONS **28** CITATIONS
SEE PROFILE

Sebastiaan van Zelst
Fraunhofer Institute for Applied Information Technology FIT
80 PUBLICATIONS **732** CITATIONS
SEE PROFILE

Wil Van der Aalst
RWTH Aachen University
1,438 PUBLICATIONS **85,059** CITATIONS
SEE PROFILE

Temporal Performance Analysis for Block-Structured Process Models in Cortado

Daniel Schuster[1,2][0000−0002−6512−9580], Lukas Schade[2], Sebastiaan J. van Zelst[1,2][0000−0003−0415−1036], and Wil M. P. van der Aalst[1,2][0000−0002−0955−6940]

[1] Fraunhofer Institute for Applied Information Technology, Sankt Augustin, Germany
{daniel.schuster,sebastiaan.van.zelst}@fit.fraunhofer.de
[2] RWTH Aachen University, Aachen, Germany
lukas.schade@rwth-aachen.de, wvdaalst@pads.rwth-aachen.de

Abstract. Process mining techniques provide insights into operational processes by systematically analyzing event data generated during process execution. These insights are used to improve processes, for instance, in terms of runtime, conformity, or resource allocation. Time-based performance analysis of processes is a key use case of process mining. This paper presents the performance analysis functionality in the process mining software tool Cortado. We present novel performance analyses for block-structured process models, i.e., hierarchical structured Petri nets. By assuming block-structured models, detailed performance indicators can be calculated for each block that makes up the model. This detailed temporal information provides valuable insight into the process under study and facilitates analysts to identify optimization potential.

Keywords: Process mining · Performance analysis · Alignments

1 Introduction

Process mining [1] comprises various methods to systematically analyze event data that are generated during the execution of operational processes and stored within organizations' information systems. Analyzing event data offers great potential for gaining valuable insights into the process under investigation. These insights are used to improve processes, i.e., the key objective of process mining.

The temporal performance analysis of processes is of major practical relevance, e.g., to determine bottlenecks within processes. This paper presents novel performance analysis functionality for *block-structured process models* in the tool Cortado [11]. By focusing on block-structured models, performance indicators (PIs) can be calculated for each block that makes up the process model individually. Calculating PIs for individual blocks of block-structured models represents a novelty compared to existing tools, which often only offer performance analysis per activity in the process model and globally, i.e., for the entire process model. In short, Cortado contributes to the state-of-the-art in model-based performance analysis.

The subsequent sections are structured as follows. Sect. 2 introduces preliminaries. Sect. 3 presents Cortado's model-based performance analysis approach

Table 1: Example of an event log

Event ID	Case ID	Activity label	Start Timestamp	Completion Timestamp	...
1	1	activity A	07/13/21 08:00	07/13/21 09:30	...
2	1	activity B	07/13/21 08:30	07/13/21 11:00	...
3	1	activity C	07/13/21 09:00	07/13/21 12:00	...
4	1	activity D	07/13/21 11:30	07/13/21 13:30	...
5	1	activity E	07/13/21 11:40	07/13/21 13:00	...
6	1	activity B	07/13/21 14:30	07/13/21 16:00	...
7	1	activity B	07/13/21 16:30	07/13/21 17:00	...
8	2	activity A	07/13/21 08:00	07/13/21 09:30	...
9	2	activity B	07/13/21 09:00	07/13/21 10:00	...
⋮	⋮	⋮	⋮	⋮	

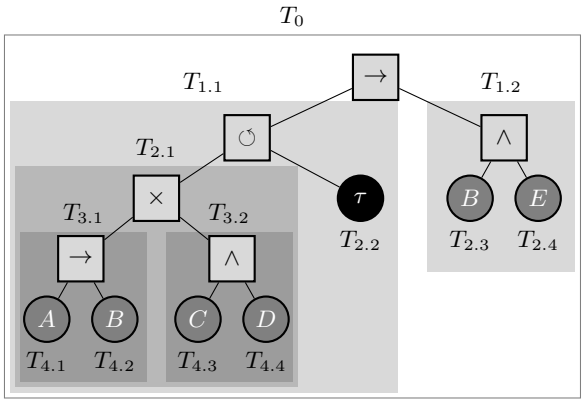


Fig. 1: Example of a process tree T_0. Subtrees are highlighted in gray

including various PIs calculated for block-structured models. Sect. 4 presents related work and tools. Finally, Sect. 5 concludes this paper.

2 Preliminaries

Event data, as considered in process mining [1], describe recorded process executions. Table 1 shows an example of an event log. Each row corresponds to an event capturing the execution of a process activity. Each event is assigned to a case by a case ID. For instance, the first event in Table 1 shows that activity A was executed on 07/13/21 from 08:00 to 09:30 for case 1. Events assigned the same case-id are also referred to as a *trace*.

Cortado uses *process trees* representing *block-structured* process models that are a subclass of sound Workflow-nets (WF-nets). Fig. 1 shows an example tree T_0. Each inner node, including the root node, represents an operator that specifies the control-flow of its children. Four operators exist: sequence ($\rightarrow$), choice ($\times$), loop ($\circlearrowleft$), and parallelism ($\wedge$). Leaf nodes represent process activities or the so-called silent activity τ. Fig. 2 shows the corresponding WF-net, describing the same language as process tree T_0. We use silent transitions (cf. black filled

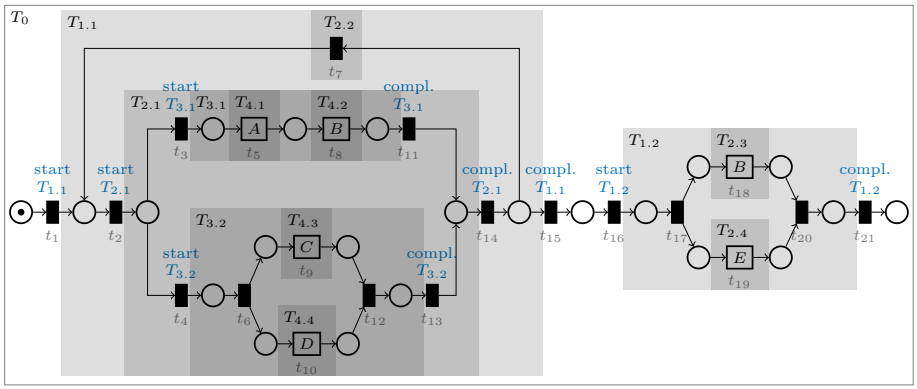


Fig. 2: Workflow net corresponding to process tree T_0. Silent transitions are used to represent the start and completion of the corresponding blocks, i.e., subtrees.

$\gg$	$\gg$	$\gg$	A	B	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	D	C	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	E	E	$\gg$	$\gg$	$\gg$
τ	τ	τ	A	B	τ	τ	τ	τ	τ	τ	D	C	τ	τ	τ	τ	τ	τ	E	$\gg$	B	τ	τ
t_1	t_2	t_3	t_5	t_8	t_{11}	t_{14}	t_7	t_2	t_4	t_6	t_{10}	t_9	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{19}		t_{18}	t_{20}	t_{21}

Fig. 3: Optimal alignment for $\langle A, B, D, C, E, E \rangle$ and the WF-net from Fig. 2

transitions in Fig. 2) to represent the start and completion for each subtree that is not a leaf node. For a formal introduction to process trees and the translation to and from WF-nets and process trees, we refer to [1].

Alignments [2] relate observed process behavior (i.e., event data) with modeled behavior (i.e., a process model). Fig. 3 shows an alignment for the trace $\langle A, B, D, C, E, E \rangle$ and the WF-net from Fig. 2. The first row of an alignment always corresponds to the given trace, ignoring the skip symbol $\gg$. The second row always corresponds to a valid firing sequence from the initial to the final marking. Each column represents a move; we distinguish four types: **synchronous moves** indicate a synchronization between the model and the trace, **log moves** indicate a *deviation*, i.e., the current activity in the trace is not replayed in the model, **visible model moves** indicate a *deviation*, i.e., the model executes an activity not observed in the trace at this stage, and **invisible model moves** indicate *no* real deviation, i.e., a model move on a transition labeled with τ. An alignment is *optimal* if it minimizes log moves and visible model moves. Note that multiple optimal alignments may exist for a given trace and a WF-net.

3 Model-Based Performance Analysis

This section introduces the model-based performance analysis approach in Cortado. The remainder is structured as follows. Sect. 3.1 introduces various performance indicators. Sect. 3.2 presents the implementation in Cortado from a user's perspective. Finally, Sect. 3.3 outlines the calculation and discusses open challenges.

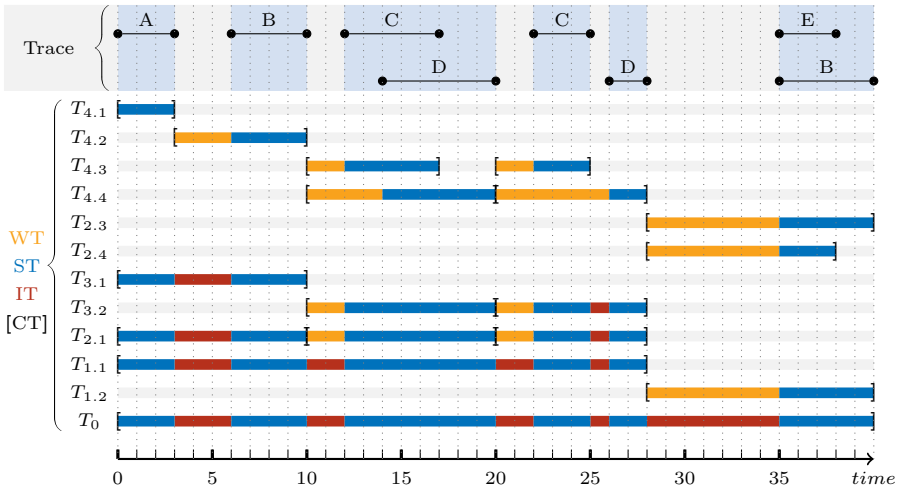
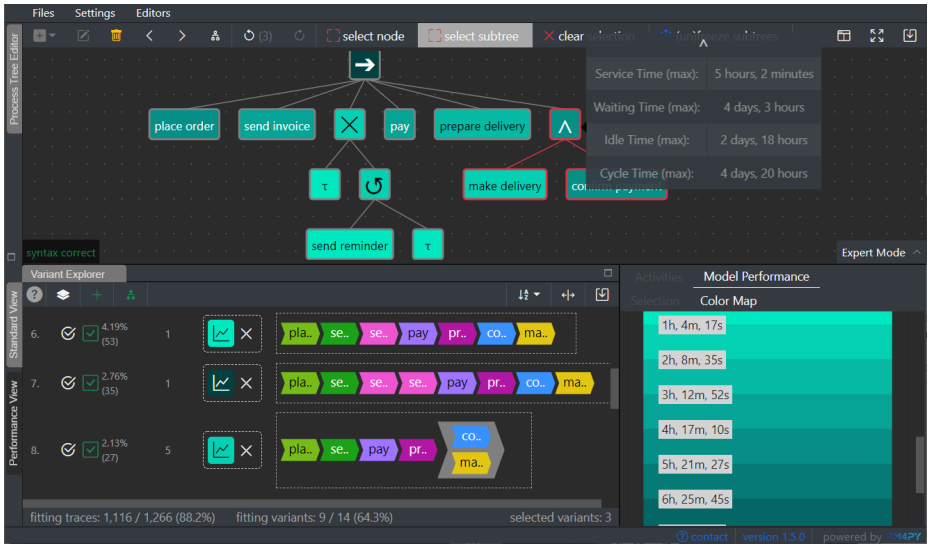


Fig. 4: Example of the PIs cycle (CT), service (ST), waiting (WT), and idle time (IT) based on a *fitting* trace for process tree T_0 (cf. Fig. 1).

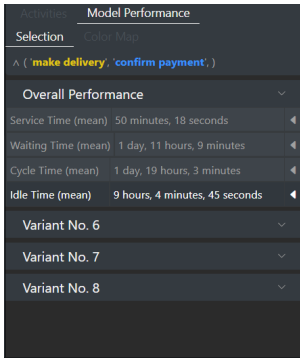
3.1 Defining Performance Indicators

Fig. 4 depicts an example of the PIs computed by Cortado: waiting time (WT), service time (ST), idle time (IT), and cycle time (CT). The PIs are calculated for a given trace and process tree T_0 (cf. Fig. 1). At the top of Fig. 4, we show the trace, consisting of eight activities visualized as intervals. Below, we show the PIs for each subtree. For instance, the first row describes $T_{4.1}$. The [symbol indicates that $T_{4.1}$ was started at time 0 and the symbol] indicates its completion at time 3. After executing $T_{4.1}$, $T_{4.2}$ must be executed according to the process model, cf. Fig. 1. Thus, $T_{4.2}$ is enabled directly after the completion of $T_{4.1}$ at time 3. Since the activity B starts at time 6 according to the given trace, the waiting time of $T_{4.2}$ is 3. The leaf nodes $T_{4.1}$ and $T_{4.2}$ belong both to the subtree $T_{3.1}$. Thus, the cycle time of $T_{3.1}$ is 10, from the start of activity A at time 0 to the end of activity B at time 10. The service time of $T_{3.1}$ is 7, i.e., the union of the service times of its leaf nodes. The waiting time of $T_{3.1}$ is 0 because after the activation of $T_{3.1}$ at time 0, the activity A was directly executed. Finally, the idle time of $T_{3.1}$ is 3, which corresponds to the waiting time of $T_{4.2}$.

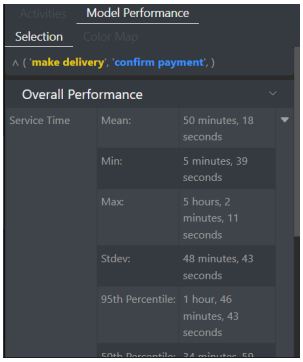
The idle time of leaf nodes is always zero because an activity cannot be paused since we only consider the start and completion of individual activities. The waiting time of an inner node corresponds to the time that elapses from its activation to the activation of its first executed leaf node. Note that the root node's waiting time is always zero since it is immediately activated when the first activity from the trace is executed. The root node is immediately closed when the last activity is executed. Once a subtree is active and a first leaf node of this subtree has already been executed, periods in which no leaf node of the subtree is being executed are considered idle time. The execution of invisible leaf nodes happens instantly and does not cause any waiting, service, or idle time.



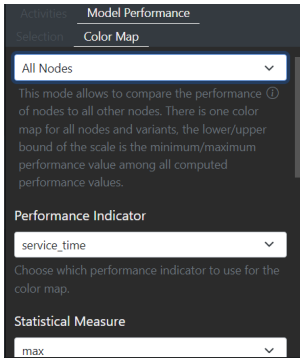
(a) Mapping max service times onto the process tree using traces corresponding to the selected variants 6., 7., and 8.



(b) PIs for the selected subtree per variant and overall.



(c) Selected subtree's service time for all variants.



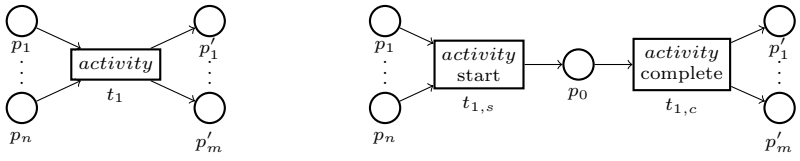
(d) Color map settings.

Fig. 5: Example of the model-based performance functionality in Cortado

For example, between closing $T_{3.1}$ and opening $T_{3.2}$ at time 10 (cf. Fig. 4) the invisible leaf node $T_{2.2}$ is executed (cf. Fig. 1). However, this execution is not visualized in Fig. 4 as it is irrelevant for the PIs. Finally, the cycle time of any (sub-)tree is the sum of its waiting, service, and idle times.

3.2 Realization in Cortado

We refer to [11] for a general introduction to the tool and details on Cortado's architecture. Cortado is available as a standalone build and can be downloaded from https://cortado.fit.fraunhofer.de. Fig. 5 shows screenshots of the model-based performance analysis functionality. Fig. 5a shows the entire User Interface



(a) Initial case/before splitting (b) After splitting

Fig. 6: WF-net preparation for performance analysis—splitting visible transitions

(UI). The trace variants from the event log are visualized in the lower left UI-component, called the variant explorer. All shown variants in Fig. 5a are fitting the process tree, which is indicated by the green check-mark to the left of each variant. Variants 6, 7, and 8 are used for the model-based performance analysis because their performance analysis button is non-gray. As such, a user can individually select variants, i.e., traces corresponding to the variants, that should be used for performance analysis by pressing the performance analysis button. In the lower right UI-component (cf. Fig. 5a), a color map is displayed that currently shows the max service times in the tree. Note the subtree selected by the user highlighted in red (cf. Fig. 5a). The same UI-component displaying the color map (cf. Fig. 5a) also provides performance statistics for the selected subtree (cf. Fig. 5b). Cortado provides the PIs for this subtree either per variant or for all selected variants, i.e., overall performance (cf. Fig. 5). Fig. 5c shows the service time statistics for the overall performance (all incorporated variants, i.e., 6, 7, and 8) for the selected subtree. As shown in Fig. 5d, the color map can be fully configured; all four PIs can be shown. Further, Cortado offers a second mode called *Variant Comparison* next to mode *All Nodes* (Fig. 5d), currently selected. The *Variant Comparison* mode allows comparing the performance of a variant to all other variants. Therefore, a user must select a single variant in the variant explorer by clicking on the performance button. The color map then shows how the performance of the selected variant compares to all other variants.

3.3 Calculating PIs and Dealing With Non-Fitting Traces

This section briefly outlines the computation of the presented PIs. First, we convert a given process tree into a WF net, as exemplified in Fig. 2. Next, we split each non-silent transition, which represent a process activity, into two transitions indicating the start and the completion of the related activity. Fig. 6 illustrates this splitting. A non-silent transition t_1 labeled *activity* is split into two transitions $t_{1,1}$, representing the start of *activity*, and $t_{1,2}$, representing its completion. Next, we calculate an optimal alignment [2] for each trace using the prepared WF-net. Since the alignment provides a full execution sequence throughout the model, we can replay the trace and track the transitions' execution timestamps.

In general, the reliability of model-based performance analysis depends on the quality of the used process model, i.e., how accurately it represents reality [1, Section 3.3.3]. Thus, the significance of performance analysis is low if the event log and a model have little behavior in common. However, non-fitting traces,

Table 2: Overview of calculable PIs for a process tree per alignment combination representing the start and completion of an activity

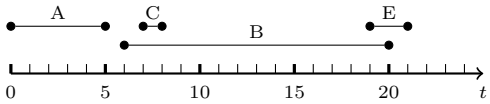
#	Alignment move combinations	Interpretation	WT	ST	IT	CT
1	Synchronous move on start & complete	Perfect activity instance	✓	✓	✓	✓
2	Synchronous move on start & model move on complete	Partial start	✓	-	✓[a]	-
3	Model move on start & synchronous move on complete	Partial complete	✓[b]	-	✓[a]	✓[c]
4	Model move on start & complete	Missing activity instance as per model	-	-	-	-
5	Log move	Missing activity instance as per log	-	-	-	-

[a] Cannot be used for the IT of the actual activity instance since it cannot be paused. However, the information on start resp. completion can be potentially used to determine the IT for blocks/subtrees containing this activity.
[b] Cannot be used for the WT of the actual activity instance. However, the completion information can be used to determine the WT for a subsequently executed activity.
[c] Cannot be used for the CT of the actual activity instance. Instead, the completion information can be potentially used to determine the CT for blocks/subtrees containing this activity.

i.e., traces not entirely replayable on a process model, can be incorporated for performance analysis, cf. [2]. Also, Cortado allows the utilization of non-fitting traces for performance analysis.

Table 2 provides an overview of different situations, i.e., combinations of alignment moves for a single activity instance, and the corresponding possibilities regarding their incorporation into the calculation of PIs. For instance, the first alignment move combination is a synchronous move on an activity's start and completion. This combination is called a perfect activity instance, i.e., we could replay the start and the completion of an activity in the model as it occurred in the event data. Reconsider the example in Fig. 4 where each activity from the trace corresponds to a perfect activity instance. We can utilize the information from a perfect activity instance for all four PIs. In the second combination, we only know when an activity started, i.e., a synchronous move on the activity's start. However, we do not know precisely when it was completed as we observe a model move on complete. In this case, we can utilize the timing information from the synchronous move to calculate waiting times and cycle times of corresponding blocks and the entire process model. The third case describes a partial complete. This case is particularly interesting because many event logs do not contain two timestamps for each process activity in practice. Instead, only completion timestamps are available. Under these circumstances, we can still compute waiting, idle, and cycle times, cf. Table 2. Finally, note that we cannot use the alignment information for any PI calculation in the fourth and fifth cases.

Although non-fitting traces can be incorporated for performance analysis (cf. Table 2), the nature of alignments—multiple optimal alignments may exist for a given trace and model—adds a randomness factor to the performance analysis. A detailed examination of this problem is outside this paper's scope; instead, Fig. 7 illustrates an example of the problem. Fig. 7b shows different optimal alignments for a trace (cf. Fig. 7a) and the WF-net from Fig. 2 after splitting transitions (cf. Fig. 6). Using these different alignments to compute the service



$\langle A$ start, A complete, B start, C start, C complete, E start, B complete, E complete$\rangle$

(a) Non-fitting trace and its sequential representation, used for alignment calculation

A start	A complete	B start	C start	$\gg$	$\gg$	C complete	E start	B complete	E complete
A start $t_{5,s}$	A complete $t_{5,c}$	B start $t_{7,s}$	$\gg$	B complete $t_{7,s}$	B start $t_{18,s}$	$\gg$	E start $t_{19,s}$	B complete $t_{18,c}$	E complete $t_{19,c}$
0	5	6	-	-	-	-	19	20	21

Service time of the root node: $\boldsymbol{ST(T_0) = (5-0) + (21-19) = 7}$

$\boldsymbol{A}$ start	$\boldsymbol{A}$ complete	$\boldsymbol{B}$ start	$\gg$	$\gg$	$\boldsymbol{C}$ start	$\gg$	$\boldsymbol{C}$ complete	$\gg$	$\boldsymbol{E}$ start	$\boldsymbol{B}$ complete	$\boldsymbol{E}$ complete
$\boldsymbol{A}$ start $t_{5,s}$	$\boldsymbol{A}$ complete $t_{5,c}$	$\boldsymbol{B}$ start $t_{7,s}$	$\boldsymbol{B}$ complete $t_{7,s}$	$\boldsymbol{D}$ start $t_{10,s}$	$\boldsymbol{C}$ start $t_{9,s}$	$\boldsymbol{D}$ complete $t_{10,c}$	$\boldsymbol{C}$ complete $t_{9,c}$	$\boldsymbol{B}$ start $t_{18,s}$	$\boldsymbol{E}$ start $t_{19,s}$	$\boldsymbol{B}$ complete $t_{18,c}$	$\boldsymbol{E}$ complete $t_{19,c}$
0	5	6	-	-	7	-	8		19	-	21

Service time of the root node: $\boldsymbol{ST(T_0) = (5-0) + (8-7) + (21-19) = 8}$

$\boldsymbol{A}$ start	$\boldsymbol{A}$ complete	$\gg$	$\gg$	$\boldsymbol{B}$ start	$\boldsymbol{C}$ start	$\boldsymbol{C}$ complete	$\boldsymbol{E}$ start	$\boldsymbol{B}$ complete	$\boldsymbol{E}$ complete
$\boldsymbol{A}$ start $t_{5,s}$	$\boldsymbol{A}$ complete $t_{5,c}$	$\boldsymbol{B}$ start $t_{7,s}$	$\boldsymbol{B}$ complete $t_{7,s}$	$\boldsymbol{B}$ start $t_{18,s}$	$\gg$	$\gg$	$\boldsymbol{E}$ start $t_{19,s}$	$\boldsymbol{B}$ complete $t_{18,c}$	$\boldsymbol{E}$ complete $t_{19,c}$
0	5	-	-	6	-	-	19	20	21

Service time of the root node: $\boldsymbol{ST(T_0) = (5-0) + (21-6) = 20}$

(b) Optimal alignments for the given trace and the WF-net from Fig. 2 after preparation (cf. Fig. 6). For simplicity, invisible model moves are omitted. Below each synchronous move, we show the respective timestamp. Perfect instances (cf. Table 2) are colored.

Fig. 7: Example of a non-fitting trace that causes unreliable performance analysis results, e.g., the service time of T_0 depends on which optimal alignment is used

time of the process tree root, i.e., $ST(T_0)$, we end up with three different values because each alignment finds different perfect activity instances (cf. Table 2). Since only perfect activity instances can be used for service time calculation (cf. Table 2), the service time depends on the optimal alignment found. Cortado, therefore, warns the user if non-fitting traces are used for performance analysis.

4 Related Work and Tools

This section provides a brief overview of related work and tools to highlight the differences in the proposed approach. Note that an extensive and complete overview is outside this paper's scope. The fundamental idea of analyzing the temporal performance of a process and enriching process models with performance statistics has been discussed in [1, Chapter 9], [6, Chapter 7], and [4, Chapter 10]. In [2, Chapter 9], the author describes how alignments can be utilized for performance analysis for Petri nets. The usage of alignments for the performance analysis of YAWL models has been shown in [3].

We surveyed commercial and academic tools: ABBYY Timeline, Apromore [7], ARIS Process Mining, Celonis, Disco, IBM Process Mining, Minit, and the

ProM [5] plugins: Inductive Visual Miner [9], Replay Log in YAWL net [10], Discover using the Statechart Workbench [8], and Replay a Log on Petri Net for Performance Analysis [2]. All commercial tools and the Inductive Visual Miner provide performance analysis for Directly-Follows-Graphs (DFG). A DFG is a directed graph representing the directly follows relation of process activities. The expressiveness of a DFG is, however, limited [12] compared to the expressiveness of, e.g., process trees. Performance analysis for non-DFG models is only supported by Apromore, ARIS, and the ProM plugins. The plugin Discover using the Statechart Workbench [8] also supports hierarchical performance analysis of process trees as Cortado, but it only offers cycle time statistics per block. In contrast, Cortado's approach calculates various PIs for each block.

5 Conclusion

This paper presented the performance analysis functionality of Cortado. Focusing on block-structured models, PIs—we presented cycle, idle, waiting, and service time—can be calculated for each block, each of which represents a part of the overall process model. The main challenges for future work remain the scalability of the alignment computation and the highlighted problem of incorporating non-fitting traces in model-based performance analysis.

References

1. van der Aalst, W.M.P.: Data Science in Action. Springer (2016). https://doi.org/10.1007/978-3-662-49851-4_1
2. Adriansyah, A.: Aligning observed and modeled behavior. Ph.D. thesis (2014). https://doi.org/10.6100/IR770080
3. Adriansyah, A., Van Dongen, B., Piessens, D., Wynn, M., Adams, M.: Robust performance analysis on yawl process models with advanced constructs. Journal of Information Technology Theory and Application (JITTA) **12**(3) (2012). https://doi.org/10.1.1.227.6079
4. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018). https://doi.org/10.1007/978-3-319-99414-7
5. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The prom framework: A new era in process mining tool support. In: Applications and Theory of Petri Nets. Springer (2005). https://doi.org/10.1007/11494744_25
6. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer (2013). https://doi.org/10.1007/978-3-642-33143-5
7. La Rosa, M., Reijers, H.A., van der Aalst, W.M., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L.: Apromore: An advanced process model repository. Expert Systems with Applications **38**(6) (2011). https://doi.org/10.1016/j.eswa.2010.12.012
8. Leemans, M., van der Aalst, W.M.P., van den Brand, M.G.J.: Hierarchical performance analysis for process mining. Association for Computing Machinery (2018). https://doi.org/10.1145/3202710.3203151
9. Leemans, S.: Robust process mining with guarantees. Ph.D. thesis (2017)
10. Piessens, D., Wynn, M.T., Adams, M., van Dongen, B.F., et al.: Performance analysis of business process models with advanced constructs (2010)

11. Schuster, D., van Zelst, S.J., van der Aalst, W.M.P.: Cortado—an interactive tool for data-driven process discovery and modeling. In: Application and Theory of Petri Nets and Concurrency. Springer (2021). https://doi.org/10.1007/978-3-030-76983-3_23
12. van der Aalst, W.M.: A practitioner's guide to process mining: Limitations of the directly-follows graph. Procedia Computer Science **164** (2019). https://doi.org/10.1016/j.procs.2019.12.189