

# Kronos: Discovery and Analysis of Waiting Time Causes

Katsiaryna Lashkevich\*, Fredrik Milani, David Chapela-Campa, Ihar Suvorau and Marlon Dumas

University of Tartu, 18 Narva mnt, Tartu, 51009, Estonia

## Abstract

Waiting times often occur in business processes when a case transitions from one activity to another. Accordingly, analyzing the causes of waiting times in activity transitions can aid analysts in identifying opportunities for reducing process cycle time. To solve this task, we propose Kronos – a web-based tool that decomposes waiting times in activity transitions into their causes and analyzes their impact on the cycle time efficiency of the process. Thus, Kronos targets process analysts interested in optimizing the temporal efficiency of business processes.

## Keywords

process mining, waiting time, cycle time efficiency

## 1. Introduction

Waiting time is a common source of waste in business processes [1]. Waiting times typically arise during transitions between activities, i.e., when the execution of a case moves from one activity to another. Process analysts benefit from understanding what causes waiting times when exploring how to address such process inefficiencies.

Process mining techniques enable analysis of data generated by business process executions, a.k.a. *event logs*, and, in particular, to discover waiting times [2]. However, while existing techniques enable analysts to visualize activity transitions with high waiting time (i.e., bottlenecks), they provide limited support for identifying causes of waiting times.

In this paper, we present Kronos, an open-source web-based tool that discovers the causes of waiting times in activity transitions. Kronos also assesses the impact each cause of waiting time has on the process's cycle time efficiency (CTE). This can aid analysts in identifying improvement opportunities related to waiting times that, when addressed, can increase the temporal efficiency of the process.

The rest of the paper is structured as follows. Sec. 2 describes Kronos's functionality and components. Sec. 3 presents the maturity and availability of Kronos. Sec. 4 concludes the paper.

---

ICPM 2023 Doctoral Consortium and Tool Demonstration Track, October 23–27, 2023, Rome, Italy

\*Corresponding author.


✉ katsiaryna.lashkevich@ut.ee (K. Lashkevich); fredrik.milani@ut.ee (F. Milani); david.chapela@ut.ee (D. Chapela-Campa); ihar.suvorau@ut.ee (I. Suvorau); marlon.dumas@ut.ee (M. Dumas)

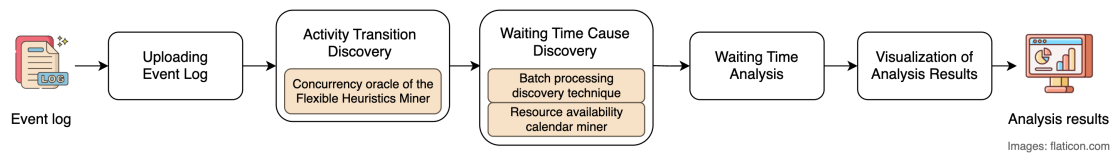
🆔 0000-0003-4426-7738 (K. Lashkevich); 0000-0002-1322-915X (F. Milani); 0000-0002-4711-9653

(D. Chapela-Campa); 0000-0002-1862-2604 (I. Suvorau); 0000-0002-9247-7476 (M. Dumas)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)



**Figure 1:** Overview of Kronos's structure.

## 2. Architecture & Main Features

Kronos takes an event log as input, identifies transitions, calculates their durations (waiting times), discovers and quantifies the causes of the waiting times, measures their impact on the process CTE and, finally, visualizes the results. Figure 1 gives an overview of Kronos's components. Below, we summarize the functionality of each component of Kronos.

### 2.1. Uploading Event Log

As input, Kronos takes an event log (CSV format) with at least a unique case identifier, activity name, resource, start and end timestamps. These attributes are required. When the log is uploaded, the user maps the columns to their respective attributes, and Kronos validates that all mandatory attributes are included in the log. If so, Kronos proceeds to the activity transition discovery.

### 2.2. Activity Transition Discovery

In this component, Kronos identifies activity transitions, i.e., pairs of activities – composed of a source and a target activity, where the source activity enables the execution of the target activity – between which cases are transferred. With this purpose, Kronos first discovers the concurrency relations between the process activities (using the concurrency oracle of the Flexible Heuristics Miner [3]). Then, it builds the activity transitions by pairing each activity instance with its preceding non-concurrent one, i.e., the activity instance enabling it. Finally, Kronos calculates the duration of each transition – i.e., the waiting times they induce – as the time from the end of its source activity to the start of its target one.

### 2.3. Waiting Time Cause Discovery

Once the activity transitions and their waiting times are discovered, Kronos discovers the causes of waiting times. The waiting time within a given transition instance may stem from one or multiple causes. If there are multiple causes, Kronos decomposes the waiting time into non-overlapping time intervals and attributes each interval to one cause. Kronos identifies five causes of waiting time in the following order:

- *Waiting time due to batching* occurs when an activity instance waits for another activity instance to be enabled in order to be processed together as a batch. To identify batch processing, Kronos uses the technique proposed in [4].

- *Waiting time due to resource contention* is observed when an activity instance waits to be processed by an assigned resource that is busy processing other activity instances, following a first-in-first-out (FIFO) order.
- *Waiting time due to prioritization* is identified when the assigned resource is busy with an activity instance that was prioritized over the waiting one (not executed in the FIFO order).
- *Waiting time due to resource unavailability* occurs when the assigned resource is unavailable (off duty) due to their working schedules. Kronos discovers the working schedules of each resource using the resource availability calendar miner proposed in [5].
- *Waiting time due to extraneous factors* covers waiting times caused by external effects that cannot be identified from the event log – e.g., the resource is working on another process, fatigue effects, or context switches.

The order by which waiting time causes are identified in a given activity transition is determined by the dominance relations between these causes. Batching dominates resource contention, prioritization, and unavailability, because regardless of the availability status of any given resource, an activity instance that is part of a batch is not ready to be assigned (and started) until the batch is ready. Resource contention and prioritization dominate resource availability, because if a resource has a work queue, they cannot start an activity instance until the latter reaches the front of the queue, or until this activity instance has the highest priority in the queue, regardless of the resource’s availability status. Extraneous factors are dominated by all other causes, as they act as a “catch-all” cause for any waiting time that cannot be attributed to other causes.

## 2.4. Waiting Time Analysis

In this component, Kronos analyzes how much each cause of waiting time contributes to the temporal performance of the process (i.e., the percentage of time each cause induces in the process) and their impact on the CTE. The impact of each cause of waiting time is calculated as the difference between the original process CTE and the CTE if the waiting time is eliminated. In this way, Kronos measures (1) the impact each waiting time cause has on the process CTE, (2) the impact each transition has on the process CTE, and (3) the impact each waiting time cause has in each transition. These metrics can indicate the potential CTE improvement if a particular cause of waiting time is eliminated.

## 2.5. Visualization of Analysis Results

Finally, Kronos visualizes the analysis results in its user interface that has 3 tabs: (1) *Overview tab* presents the key statistics of the process (e.g., number of cases, activities, and transitions), total waiting time of the process, and how much each cause induces; (2) *Transitions tab* shows the waiting time causes per transition; (3) *CTE impact tab* depicts potential CTE improvement if the waiting time causes are eliminated in the whole process and per activity transition. Figure 2 illustrates an example of a real-life production process, where the graph shows waiting time causes per activity transition. The analysis results can be downloaded in CSV and JSON formats.

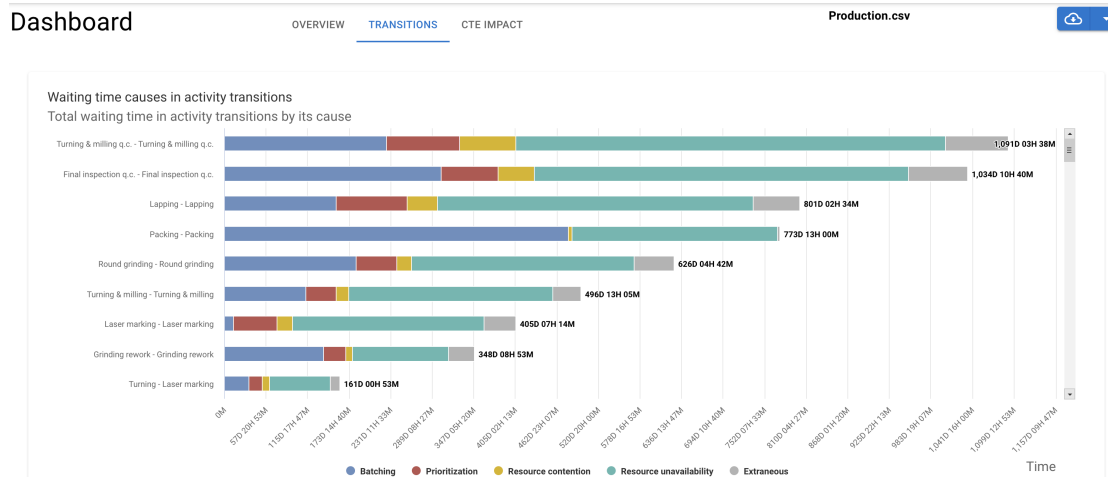


Figure 2: Waiting time causes per activity transition in an example of a real-life production process.

### 3. Maturity & Availability

Kronos has been empirically evaluated with synthetic event logs where the causes of waiting time were known [6]. The empirical evaluation showed that Kronos can accurately detect waiting times and classify them into five causes.

Kronos is developed as a React web application, publicly available at <http://kronos.cloud.ut.ee>. The current server deployment accepts event logs with sizes up to 30 MB. A set of event logs for testing is available at Owncloud<sup>1</sup>. The implementation of Kronos's logic is available in a GitHub repository<sup>2</sup>, along with instructions for its installation and command-line usage. A screencast that describes the tool is available on YouTube<sup>3</sup>.

In its current form, Kronos has several limitations in terms of method and implementation:

- *Method limitations.* First, the method considers only waiting times in transitions between activity instances. Yet waiting times may also arise in at least two other settings: (i) between case creation and the start of the first activity instance; and (ii) within an activity instance due to interruptions (e.g., the resource interrupts their work and resumes it later). The first of these waiting times could be analyzed by applying methods that estimate the inter-arrival time of each case [7]. The second requires new methods for modeling and inferring interruptions. Another limitation of the method is that it does not consider multitasking. This could be addressed by inferring multitasking patterns from the log, and using this data to estimate at what point in time a resource would normally have started an activity instance, given their past multitasking behavior. Finally, the values of potential CTE improvement depict a theoretical improvement achieved by eliminating specific waiting times, without accounting for any potential side effects resulting from a

<sup>1</sup><https://owncloud.ut.ee/owncloud/s/rZ4dSoTzwpwfpqi>

<sup>2</sup><https://github.com/AutomatedProcessImprovement/waiting-time-analysis>

<sup>3</sup>[https://youtu.be/vvOY\\_hbOOh4](https://youtu.be/vvOY_hbOOh4)

process redesign. This could be addressed by employing a simulation-based analysis to explore various redesign options considering associated side effects.

- *Implementation limitations.* Kronos has limited capacity for processing extensive event logs due to the prolonged processing time they require. Furthermore, the likelihood of encountering errors is higher for larger logs, while Kronos has limited error-handling support. Therefore, we have implemented an event log size limit of 30 MB. It allows addressing the aforementioned challenges while maintaining the capability to process intricate event logs, such as the event log from BPI Challenge 2012 [8].

## 4. Conclusion

Kronos discovers five causes of waiting times (batching, resource contention, prioritization, resource unavailability, and extraneous factors) from event logs, assesses their impact on process CTE, and visualizes the analysis results. With Kronos, process analysts can identify improvement opportunities related to increasing temporal efficiency by reducing waiting times. In the future, we plan to address the method limitations, in particular, add a simulation-based analysis, allowing analysts to experiment with redesign options.

## Acknowledgments

Work funded by the European Research Council (PIX project).

## References

- [1] P. Delias, A positive deviance approach to eliminate wastes in business processes: The case of a public organization, *Ind. Manag. Data Syst.* (2017).
- [2] W. M. P. van der Aalst, *Process Mining: Data Science in Action*, 2nd ed., Springer, Heidelberg, 2016.
- [3] A. J. M. M. Weijters, J. T. S. Ribeiro, Flexible heuristics miner (FHM), in: *Proceedings of the IEEE Symposium on CIDM*, IEEE, 2011.
- [4] K. Lashkevich, F. Milani, D. Chapela-Campa, M. Dumas, Data-driven analysis of batch processing inefficiencies in business processes, in: *RCIS*, Springer, 2022, pp. 231–247.
- [5] O. López-Pintado, M. Dumas, Business process simulation with differentiated resources: Does it make a difference?, in: *BPM*, Springer, 2022, pp. 361–378.
- [6] K. Lashkevich, F. Milani, D. Chapela-Campa, I. Suvorau, M. Dumas, Why am i waiting? data-driven analysis of waiting times in business processes, in: *CAiSE*, Springer, 2023, pp. 174–190.
- [7] N. Martin, B. Depaire, A. Caris, Using event logs to model interarrival times in business process simulation, in: *Workshops of the 13th Intl. Conf. on BPM*, Springer, 2015, pp. 255–267.
- [8] B. van Dongen, Bpi challenge 2012, 2012. URL: [https://data.4tu.nl/articles/\\_/12689204/1](https://data.4tu.nl/articles/_/12689204/1). doi:10.4121/UUID:3926DB30-F712-4394-AEBC-75976070E91F.