Cortado — An Interactive Tool for Data-Driven Process Discovery and Modeling

Daniel Schuster[1][0000−0002−6512−9580], 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 FIT, Germany
{daniel.schuster,sebastiaan.van.zelst}@fit.fraunhofer.de
[2] RWTH Aachen University, Germany
wvdaalst@pads.rwth-aachen.de

Abstract. Process mining aims to diagnose and improve operational processes. Process mining techniques allow analyzing the event data generated and recorded during the execution of (business) processes to gain valuable insights. Process discovery is a key discipline in process mining that comprises the discovery of process models on the basis of the recorded event data. Most process discovery algorithms work in fully automated fashion. Apart from adjusting their configuration parameters, conventional process discovery algorithms offer limited to no user interaction, i.e., we either edit the discovered process model by hand or change the algorithm's input by, for instance, filtering the event data. However, recent work indicates that the integration of domain knowledge in (semi-)automated process discovery algorithms often enhances the quality of the process models discovered. Therefore, this paper introduces Cortado, a novel process discovery tool that leverages domain knowledge while incrementally discovering a process model from given event data. Starting from an initial process model, Cortado enables the user to incrementally add new process behavior to the process model under construction in a visual and intuitive manner. As such, Cortado unifies the world of manual process modeling with that of automated process discovery.

Keywords: Process mining · Interactive process discovery · Process trees· Block-structured workflow nets · Process modeling.

1 Introduction

Process mining techniques allow analyzing the execution of (business) processes on the basis of event data collected by any type of information system, e.g., SAP, Oracle, and Salesforce. Next to *conformance checking* and *process enhancement*, *process discovery* is one of the three main sub-disciplines in process mining [3]. Process discovery aims to learn a process model from observed process behavior, i.e., *event data*. Most process discovery algorithms are fully automated. Apart from adjusting configuration parameters of a discovery algorithm, which, for instance, can influence the complexity and quality of the resulting models, the

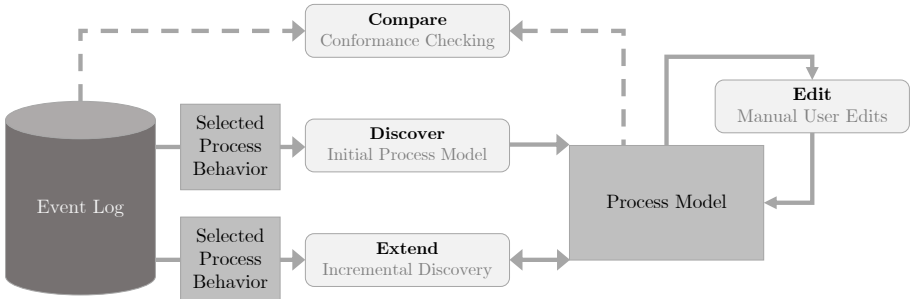


Fig. 1: Overview of Cortado's core functionality. The user discovers an initial model from user-selected process behavior. Next, the obtained process model can be incrementally extended by new process behavior from the event log. In addition, the user can edit the process model anytime and compare it with the event log

user has no direct option to steer or interact with the algorithm. Further (indirect) user interaction is limited to either changing the input, i.e., the event data fed into the discovery algorithm, or manipulating the output, i.e., the discovered process model. Thus, conventional process discovery algorithms work like a *black box* from the user's perspective.

Several studies indicate that exploiting domain knowledge within (semi-) automated process discovery leads to better process models [6,4]. Recent work has proposed the tool ProDiGy [5], allowing the user to interact with automated process discovery. However, the tool approaches the user-interaction from a *modeling-perspective*, i.e., a human modeler supported by the underlying algorithms (including an *auto-complete* option) is central to the tool and makes the design decisions for the model. Thus, model creation is still a largely manual endeavour.

This paper introduces Cortado, an interactive tool for data-driven process discovery and modeling. Cortado exploits automated process discovery to construct process models from event data in an incremental fashion. Main functionalities of our tool are visualized in Figure 1. The central idea of Cortado is the incremental discovery of a process model, which is considered to be "under construction". Cortado thereby utilizes the user's domain knowledge by delegating the decision to the user, which is about selecting the observed process behavior that gets added to the process model.

Cortado allows for discovering an initial process model from a user-selected subset of observed process behavior with a conventional process discovery algorithm (see **Discover** in Figure 1). Alternatively, one can also import a process model into Cortado. Cortado allows incrementally extending an initially given process model, which is either imported or discovered, by adding process behavior that is not yet described by the process model "under construction". Thus, the user is required to incrementally select process behavior from the event log

and to perform incremental process discovery. Our incremental discovery algorithm [10] takes the current process model and the selected process behavior and alters the process model such that the selected process behavior is described by the resulting model (see **Extend** in Figure 1). By incrementally selecting process behavior, the user *guides the incremental process discovery algorithm* by providing feedback on the correctness of the observed event data. The user therefore actively selects the process behavior to be added. Since the incremental process discovery approach allows users to undo/redo steps at any time, they have more control over the process discovery phase of the model compared to conventional approaches. To improve the flexibility of Cortado, a process model editor is also embedded, allowing the user to alter the process model at any time (see **Edit** in Figure 1). Furthermore, feedback mechanisms are implemented that notify the user of the quality of the discovered process models (see **Compare** in Figure 1).

The remainder of this paper is structured as follows. In Section 2, we briefly introduce background knowledge. In Section 3, we explain the algorithmic foundation of Cortado, i.e., the incremental process discovery approach. In Section 4, we present our tool and explain its main functionality and usage. In Section 5, we briefly describe the underlying implementation. Section 6 concludes the paper.

2 Background

In this section, we briefly explain the concept of event data and present *process trees*, which is the process modeling formalism used by Cortado.

2.1 Event Data

The information systems used in companies, e.g., Customer Relationship Management (CRM) and Enterprise Resource Planning (ERP) systems, track the performed activities during the executions of a process in great detail.

Table 1 presents a simplified example of such event data, i.e., referred to as an *event log*. Each row represents an *event*, a recording related to some *activity instance* of the process. For example, the first row indicates that a fine with identifier `A1` was created on July 24, 2006. The next line/event records that the same fine was sent. Note that the corresponding expense for sending the fine was €11.0, the Article of this violation is 157, the vehicle class is A, etc. Multiple rows have the same value for the *Fine*-column, i.e., often referred to as the *case identifier*; all these events are executed for the same instance of the process, e.g., for the same customer, the same patient, the same insurance claim, or, in the given case, for the same fine. We refer to the digital recording of a process instance as a *case*. As such, an *event log*, describes a *collection of cases*. In Cortado, we focus on *trace variants*, i.e., unique sequences of executed activities. For instance, for the fine A1 we observe the trace ⟨*Create Fine, Send Fine*⟩ and for the fine A100 ⟨*Create Fine, Send Fine, Insert Fine Notification, Add penalty, Send for Credit Collection*⟩. Note that, in general, there may be several cases for which the same sequence of activities has been performed.

Table 1: Example (simplified) event data, originating from the *Road Traffic Fine Management Process Event Log* [8]. Each row records an activity executed in the context of the process. The columns record various data related to the corresponding fine and the activity executed.

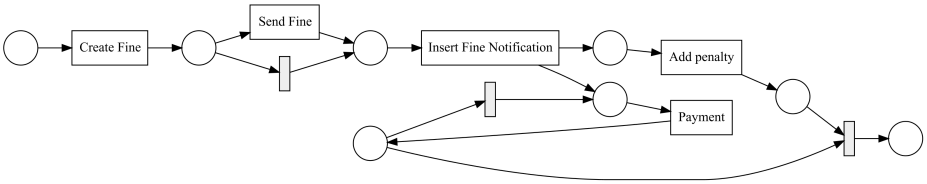
Fine	*Event*	*Start*	*Complete*	*Amount*	*Notification*	*Expense*	*Payment*	*Article*	*Vehicle Class*	*Total Payment*
A1	Create Fine	2006/07/24	2006/07/24	35.0				157	A	0.0
A1	Send Fine	2006/12/05	2006/12/05	35.0		11.0		157	A	0.0
A100	Create Fine	2006/08/02	2006/08/02	35.0				157	A	0.0
A100	Send Fine	2006/12/12	2006/12/12	35.0		11.0		157	A	0.0
A100	Insert Fine Notification	2007/01/15	2007/01/15	35.0	P	11.0		157	A	0.0
A100	Add penalty	2007/03/16	2007/03/16	71.5	P	11.0		157	A	0.0
A100	Send for Credit Collection	2009/03/30	2009/03/30	71.5	P	11.0		157	A	0.0
A10000	Create Fine	2007/03/09	2007/03/09	36.0				157	A	0.0
A10000	Send Fine	2007/07/17	2007/07/17	36.0		13.0		157	A	0.0
A10000	Add penalty	2007/10/01	2007/10/01	74.0	P	13.0		157	A	0.0
A10000	Payment	2008/09/09	2008/09/09	74.0	P	13.0	87.0	157	A	87.0
...	...	...	...	...	...	...	...	...	...	...

2.2 Process Trees

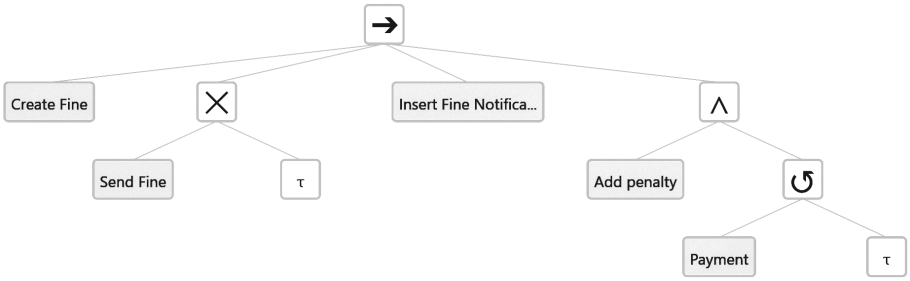
We use *process models* to describe the control-flow execution of a process. Some *process modeling formalisms* additionally allow specifying, for instance, what resources execute an activity and what data attributes in the information system might be read or written during the activity execution. In Cortado, we use *process trees* as a process modeling formalism. Process trees are a hierarchical process modeling notation that can be expressed as *sound Workflow nets* (sound WF-nets), i.e., a subclass of Petri nets, often used to model business processes. Process trees are annotated rooted trees and correspond to the class of *block-structured WF-nets*, a subclass of sound WF-nets. Process trees are used in various process discovery algorithms, e.g., the inductive miner [7].

In Figure 2, we show two simplified models of the road fine management process, which is partially shown in Table 1. Figure 2a shows a sound WF-net. Figure 2b shows a process tree describing the same behavior as the model in Figure 2a. Both models describe that the *Create Fine* activity is executed first. Secondly, the *Send Fine* activity is optionally executed. Then, the *Insert Fine Notification* activity is performed, followed by a block of concurrent behavior including *Add penalty* and potentially multiple executions of *Payment*.

The semantics of process trees are fairly simple, and, arguably, their hierarchical nature allows one to intuitively reason about the general process behavior. Reconsider Figure 2b. We refer to the internal vertices as *operators* and use them to specify control-flow relations among their children. The leaves of the tree refer to *activities*. The *unobservable activity* is denoted by τ. In terms of operators, we distinguish four different types: the *sequence operator* ($\rightarrow$), the *exclusive choice operator* ($\times$), the *parallel operator* ($\wedge$), and the *loop operator* ($\circlearrowleft$). The sequence operator ($\rightarrow$) specifies the execution of its subtrees in the given order from left to right. The exclusive choice operator ($\times$) specifies that *exactly one* of its subtrees gets executed. The parallel operator ($\wedge$) specifies that all subtrees get executed



(a) Simple example Petri net (sound WF-net) modeling a road fine management process.



(b) A process tree modeling the same behavior as the Petri net in Figure 2a.

Fig. 2: Two process models, a Petri net (Figure 2a) and a process tree (Figure 2b), describing the same process behavior, i.e., a simplified fine management process

in any order and possibly interleaved. The loop operator (↺) has exactly two subtrees. The first subtree is called the "do-part", which has to be executed at least once. The second subtree is called the "redo-part", which is optionally executed. If the redo-part gets executed, the do-part is required to be executed again.

3 Algorithmic Foundation

In this section, we briefly describe the algorithmic foundation of Cortado's incremental process discovery approach [10]. Consider Figure 3, in which we present a schematic overview on said algorithmic foundation.

As an input, we assume a process model M, which is either given initially or the result of a previous iteration of the incremental discovery algorithm. Additionally, a *trace* $\sigma' = \langle a_1, \ldots, a_n \rangle$, i.e., a sequence of executed activities $a_1, \ldots, a_n$, is given. We assume that the trace σ' is not yet part of the language of model M (visualized as $\sigma' \notin L(M)$). Note that σ' is selected by the user. If the incremental procedure has already been executed before, i.e., traces have been already added to the process model in previous iterations, we use those traces as input as well (visualized as $\{\sigma_1, \sigma_2, \sigma_3, \ldots\}$ in Figure 3). The incremental process discovery algorithm transforms the three input artifacts into a new process model M' that describes the input trace σ' and the previously added traces $\{\sigma_1, \sigma_2, \sigma_3, \ldots\}$. In

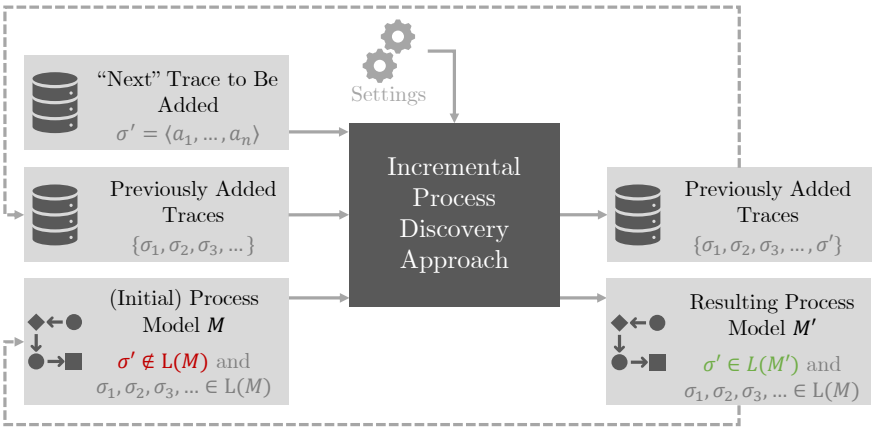


Fig. 3: Schematic overview of incremental process discovery (presented in our earlier work [10]), i.e., the algorithmic foundation of Cortado. Starting with an initial process model M and observed process behavior (a trace σ' capturing a sequence of executed process activities: $a_1, \ldots, a_n$) that is not yet captured by the model, the incremental discovery approach alters the given process model M into a new model M' that additionally accepts the given trace σ'

the next iteration, the user selects a new trace σ'' to be added and the set of previously added traces gets extended, i.e., $\{\sigma_1, \sigma_2, \sigma_3, \ldots\} \cup \{\sigma'\}$.

As mentioned before, Cortado uses process trees as a process model formalism. The incremental discovery approach [10] exploits the hierarchical structure of the input process tree M and pinpoints the subtrees where the given trace σ' deviates from the language described from the model. To identify the subtrees, the process tree is converted into a Petri net and alignments [2] are calculated. Subsequently, the identified subtrees get locally replaced, i.e., M' is a locally modified version of M.

4 Functionalities and User Interface

In this section, we present the main functionalities of Cortado. We do so along the lines of the user interface of Cortado as visualized in Figure 4.

4.1 I/O Functionalities

Cortado supports various importing and exporting functionalities, which can be triggered by the user by clicking the import/export buttons visualized in the left sidebar, see Figure 4. Cortado supports importing event data stored in the IEEE eXtensible Event Stream (XES) format [1]. Furthermore, Cortado supports importing process tree models stored as a `.ptml`-file, for instance, if an

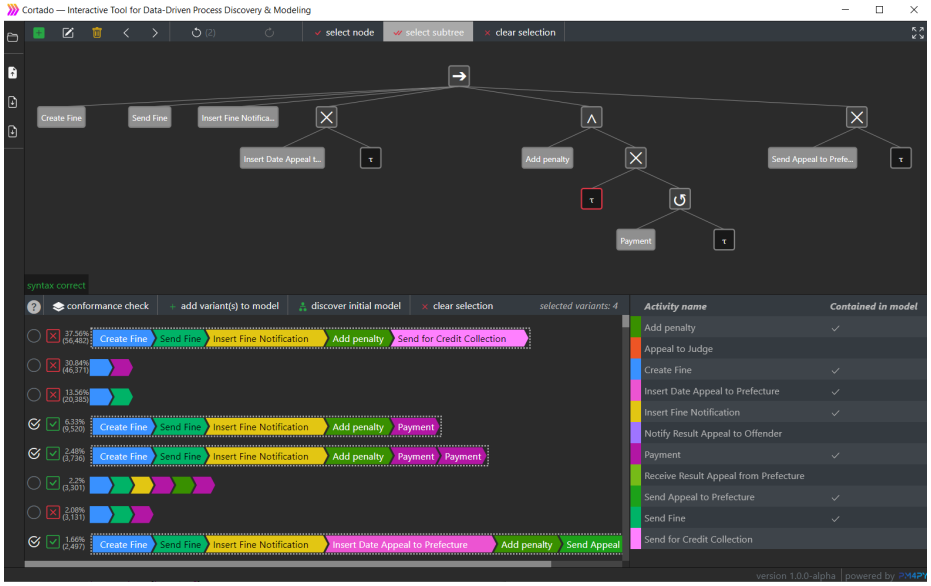


Fig. 4: Screenshot of the graphical user interface of Cortado. In the screenshot, we have loaded the *Road Traffic Fine Management Process Event Log* [8].

initial (manual) process model is available. Process tree model files (`.ptml`-files) can be generated, e.g., by process mining tools such as ProM[3] and PM4Py[4].

Next to importing, Cortado supports exporting the discovered process model both as a Petri net (`.pnml`-file) and as a process tree (`.ptml`-file). In short, Cortado offers a variety of I/0 functionalities and, hence, can be easily combined with other process mining tools.

4.2 Visualizing and Editing Process Trees

Cortado supports the visualization and editing of process trees. The "process tree under construction" – either loaded or iteratively discovered – is visualized in the upper half of the tool (Figure 4). The user can interactively select subtrees or individual vertices of the process tree by clicking an operator or a leaf node. Various edit options, e.g., removing or shifting the selected subtree left or right, are available from the top bar of the application (Figure 4). Apart from removing and shifting subtrees, the user can also add new subtrees to the process tree. Figure 5 shows a screenshot of the tree editor in detail. In the given screenshot, an inner node, a parallel operator ($\wedge$), is selected. Based on the selected inner node, the user can specify the position where to add a new node in the dropdown-menu by clicking on either `insert left`, `insert right` or `insert below`. In the

[3] https://www.promtools.org
[4] https://pm4py.fit.fraunhofer.de/

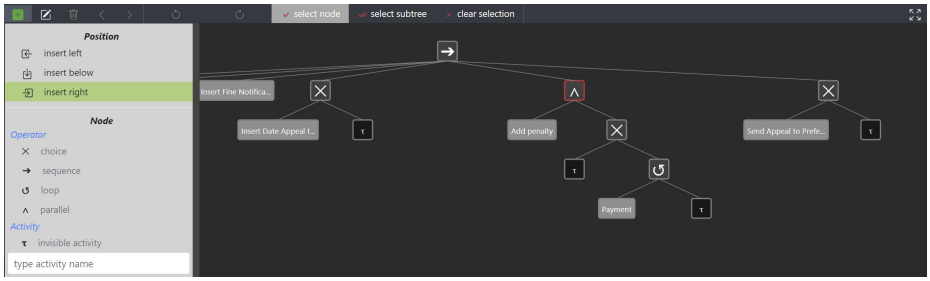


Fig. 5: Screenshot of the process tree editor in Cortado

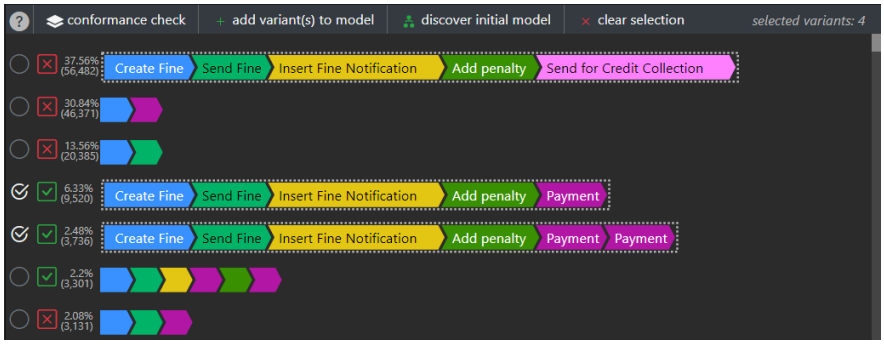


Fig. 6: Screenshot of the trace variants visualization in Cortado. There are two icons to the left of each trace variant. The left icon, a circle or a circle with a check mark, indicates whether a trace variant has been explicitly added to the model by the user. The right icon, a red cross or a green check mark, indicates if the trace variant is accepted by the current process model

given screenshot, `insert right` is selected. Next, the user can choose between an activity (a leaf node) or an operator (an inner node). By clicking on one of the options, the new node is added directly to the right of the selected node. In summary, the process tree editor in Cortado allows the user to alter the process tree at any time.

4.3 Event Data Interaction

To visualize the loaded event data, Cortado uses trace variants. Clearly, multiple instances of a process can describe the exact same behavior regarding the sequence of observed activities. For example, the most frequently observed sequence of behavior in the *Road Traffic Fine Management Process Event Log [8]* (i.e., used in Figure 4), describes the sequence: ⟨*Create Fine*, *Send Fine*, *Insert Fine Notification*, *Add Penalty*, *Send for Credit Collection*⟩. In total, the process behavior of 56,482 fines (37.56% of the total number of recorded fines) follows this sequence of activities.

Trace variants are visualized in Cortado as a sequence of *colored chevrons*. Each activity gets a unique color assigned. For instance, the activity *Create Fine* is assigned a blue color in Figure 6. Cortado sorts the trace variants based on their frequency of occurrence, descending from top to bottom. By clicking a trace variant, the user "selects a variant". Selection of multiple variants is also supported. In case an initial model does not exist, clicking the `discover initial model` button discovers one from the selected trace variants using the Inductive Miner [7], a process discovery algorithm that guarantees replay fitness on the given traces and returns a process tree. In case an initial model is present, the selected variants can be "added to the model" by clicking the `add variant(s) to model` button. In this case, Cortado performs incremental process discovery as described in Section 3.

Left to each trace variant, we see statistics about its occurrence in the event log and two icons. The left-most icon, an empty circle or a white check mark, indicates whether or not the trace variant has been explicitly added to the model by the user (Figure 6). A variant has been explicitly added by the user if either the variant was used to discover an initial model or the variant has been added to an existing model by applying incremental discovery, i.e., the variant was selected and the user pressed the button `add variant(s) to model`. Note that it is possible that a particular trace variant which was not explicitly selected by the user is described by the process model; however, after incrementally adding further variants to the model, the variant is potentially no longer described. In contrast, Cortado guarantees that explicitly added trace variants are always described by any future model incrementally discovered. However, since Cortado allows for manual tree manipulation at any time, it might be the case that an explicitly added variant is not described anymore by the tree due to manual changes to the process tree.

The right-most icon is either a red cross or a green check mark (Figure 6). These icons indicate whether a trace variant is described/accepted by the process model, i.e., if a trace variant is in the language of the process model. For the computation of these conformance statistics, we use *alignments* [2]. Therefore, we internally translate the process tree into a Petri net and execute the alignment calculation. For instance, the first three variants in Figure 6 are not accepted by the current process model, but the last two variants are accepted. Similar to triggering incremental discovery, manipulations of the process tree potentially result in variants that are no longer described by the process model. To assess the conformity of traces after editing the process tree manually, the user can trigger a new conformity check by clicking the `conformance check` button.

Lastly, Cortado shows an overview list of all activities from the loaded event log. This overview is located in the lower right part of Cortado's user interface (Figure 4). Besides listing the activity names, Cortado indicates – by using a check mark icon – which activities from the event log are already present in the process model under construction. Thereby, the user gets a quick overview of the status of the incremental discovery.

5 Implementation and Installation

The algorithmic core of Cortado is implemented in Python. For the core process mining functionality, we use the PM4Py[5] library, a python library that contains, for instance, event log handling and conformance checking functionality. The GUI is implemented using web technologies, e.g., we chose the Electron[6] and Angular[7] framework to realize a cross-platform desktop application. For the graphical representation of the process tree and the trace variants we use the JavaScript library d3.js[8].

The tool is available as a desktop application and can be freely downloaded at https://cortado.fit.fraunhofer.de/. The provided archive, a `ZIP`-file, contains an executable file that will start the tool. Upon starting, the data used within this paper, i.e., Road Traffic Fine Management Process [8], gets automatically loaded. Moreover, the archive contains examples of other event logs available as `XES`-files in the directory `example_event_logs`.

6 Conclusion and Future Work

This paper presented Cortado, a novel tool for interactive process discovery and modeling. The tool enables the user to incrementally discover a process model based on observed process behavior. Therefore, Cortado allows to load an event log and visualizes the trace variants in an intuitive manner. Starting from an initial model, which can be either imported or discovered, the user can incrementally add observed process behavior to the process model under construction. Various feedback functionalities, e.g., conformance checking statistics and the activity overview, give the user an overview of the process model under construction anytime. Supporting common file formats such as XES and PNML, Cortado can be easily used with other process mining tools.

In future work, we plan to extend Cortado's functionality in various ways. First, we aim to offer more options for the user to interact with the underlying incremental discovery approach. For example, we plan to allow the user to *lock* specific subtrees during incremental discovery to prevent these from being modified further. We also plan, in case the user changes the tree in the editor, to provide improved and instant feedback on the conformance impact the changes have w.r.t. the loaded event log and the already explicitly added trace variants. However, since the calculation of conformance checking statistics – a crucial part for instant user feedback – is computational complex, we plan to evaluate the extent to which approximation algorithms [9] can be integrated.

Next to further functionality, we plan to conduct case studies with industry partners. Thereby, we aim to focus on the practical usability of Cortado. The goal is to investigate which interaction options are meaningful and understandable for the user interacting with Cortado.

[5] https://pm4py.fit.fraunhofer.de/
[6] https://www.electronjs.org/
[7] https://angular.io/
[8] https://d3js.org/

References

1. IEEE standard for extensible event stream (XES) for achieving interoperability in event logs and event streams. IEEE Std 1849-2016 pp. 1–50 (2016), https://doi.org/10.1109/IEEESTD.2016.7740858
2. van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. WIREs Data Mining and Knowledge Discovery **2**(2), 182–192 (2012), https://doi.org/https://doi.org/10.1002/widm.1045
3. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016), https://doi.org/10.1007/978-3-662-49851-4
4. Benevento, E., Dixit, P.M., Sani, M.F., Aloini, D., van der Aalst, W.M.P.: Evaluating the effectiveness of interactive process discovery in healthcare: A case study. In: Francescomarino, C.D., Dijkman, R.M., Zdun, U. (eds.) Business Process Management Workshops - BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 362, pp. 508–519. Springer (2019), https://doi.org/10.1007/978-3-030-37453-2_41
5. Dixit, P.M., Buijs, J.C.A.M., van der Aalst, W.M.P.: ProDiGy : Human-in-the-loop process discovery. In: 12th International Conference on Research Challenges in Information Science, RCIS 2018, Nantes, France, May 29-31, 2018. pp. 1–12. IEEE (2018), https://doi.org/10.1109/RCIS.2018.8406657
6. Dixit, P.M., Verbeek, H.M.W., Buijs, J.C.A.M., van der Aalst, W.M.P.: Interactive data-driven process model construction. In: Trujillo, J., Davis, K.C., Du, X., Li, Z., Ling, T.W., Li, G., Lee, M. (eds.) Conceptual Modeling - 37th International Conference, ER 2018, Xi'an, China, October 22-25, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11157, pp. 251–265. Springer (2018), https://doi.org/10.1007/978-3-030-00847-5_19
7. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - A constructive approach. In: Colom, J.M., Desel, J. (eds.) Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7927, pp. 311–329. Springer (2013), https://doi.org/10.1007/978-3-642-38697-8_17
8. de Leoni, M., Mannhardt, F.: Road traffic fine management process (Feb 2015), https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5
9. Schuster, D., van Zelst, S., van der Aalst, W.M.: Alignment approximation for process trees. PQMI 2020 : International Workshop on Process Querying, Manipulation, and Intelligence (2020)
10. Schuster, D., van Zelst, S.J., van der Aalst, W.M.P.: Incremental discovery of hierarchical process models. In: Dalpiaz, F., Zdravkovic, J., Loucopoulos, P. (eds.) Research Challenges in Information Science. pp. 417–433. Springer International Publishing, Cham (2020), https://doi.org/10.1007/978-3-030-50316-1_25