CC Viz: A Tool for Task-Based Visualizations in Conformance Checking

Nikolaos Theofanopoulos[1,*], Michael Grohs[1,*]

[1]University of Mannheim, L15 1-6, 68161 Mannheim, Germany

Abstract

Conformance checking, a core activity in process mining, compares real-world process executions recorded in event logs to intended process behavior captured in process models. While numerous conformance checking techniques exist, their practical adoption remains limited. One contributing factor to this is that visualizations might not be suited to address the underlying user purposes. In that light, recent work has introduced a taxonomy of conformance checking tasks that captures purposes when applying conformance checking. However, current tools neither systematically evaluate their visualizations against this taxonomy nor provide integrated support for end-to-end conformance analysis. To address this gap, this paper presents CC Viz, a tool that offers five targeted visualizations, each tailored to a specific task from the taxonomy. The tasks—ranging from exploring guideline violations to presenting the impact of conformance on process outcomes—are organized as a sequential analysis pipeline. CC Viz enables users to switch between visualizations and interactively drill down into specific conformance violations, facilitating a holistic and task-aligned conformance analysis experience.

Keywords

Process Mining, Conformance Checking, Visualization

Metadata description	Value
Tool name	CC Viz
Current version	1.0
Legal code license	Apache 2.0
Languages, tools and services used	React.tsx, Python, PM4Py
Supported operating environment	Microsoft Windows, MacOS
Download/Demo URL	https://github.com/michaelgrohs/ccviz, https://ccviz-frontend.vercel.app
Documentation URL	https://github.com/michaelgrohs/ccviz/blob/master/README.md
Source code repository	https://github.com/michaelgrohs/ccviz
Screencast video	https://github.com/michaelgrohs/ccviz/blob/master/demonstration.mov

1. Introduction

Conformance checking is a sub-discipline of process mining, which compares process executions, recorded as collections of traces in an event log, to envisioned process behavior as captured in a process model [1]. In the past, many different techniques for conformance checking have been

ICPM Doctoral Consortium and Demo Track 2025, October 20-24, 2025, Montevideo, Uruguay

*Corresponding author.

nikolaos.theofanopoulos@students.uni-mannheim.de (N. Theofanopoulos); michael.grohs@uni-mannheim.de (M. Grohs)

developed. They are able to detect conformance violations whenever a trace diverges from the process model [2]. All techniques require a process model and an event log as input [1]. Based on that, users can automatically derive non-conform behavior and also directly assess which parts of traces were not desired [3].

Regardless of these capabilities, conformance checking is rarely applied in practice [4, p. 39]. This can be attributed to multiple reasons. For example, many conformance checking techniques require significant computational resources, leading to long waiting times during their usage [5]. Another reason is the reliance on process models, which may not be available in organizations since it is time-consuming and error-prone to create and maintain them [2].

In this demo, we address another reason for the lack of conformance checking usage in practice: the suitability of visualizations to address users' needs when applying conformance checking. These needs are the purpose that a user has when using conformance checking, or, in other words, the *task* the user wants to fulfill [5]. Recently, a taxonomy of conformance checking tasks has been proposed, summarizing manifold purposes of conformance checking [5]. The taxonomy consists of six dimensions:

- *task goal*: why the task is done (e.g., explore, confirm)
- *task means*: how the task is carried out (e.g., discover, compare)
- *data characteristics*: what should be revealed (e.g., conformance, guideline violations[1])
- *constraint type*: the perspective referred to (e.g., control-flow, data)
- *data target*: the data on which the task is carried out (e.g., log, trace)
- *data cardinality*: the cardinality of the data target (e.g., all, many)

Through this, users are able to communicate what they want to achieve with conformance checking. However, existing visualizations in commercial or scientific tools are not evaluated w.r.t. whether they address these tasks. Additionally, existing tools predominantly contain standalone visualizations, meaning that there is no support for a wholistic conformance analysis.

To address this gap and closely align the visualization of conformance checking results with the underlying tasks, we present the *CC Viz* tool. Given a process model and an event log as input, the tool visualizes conformance checking results that address five common tasks of the dimensions 'task goal: task means, data characteristics':

(1) Explore: Identify, Guideline violations
(2) Describe: Identify, Guideline violations
(3) Describe: Present, Conformance distribution
(4) Explain: Discover, Reasons for guideline violations
(5) Present: Compare, Impact of conformance on process outcome

We use these tasks as they can be considered a sequential analysis pipeline for conformance checking, starting with task (1) and ending with task (5) [5]. We abstract from the remaining dimensions *constraint type*, *data target*, and *data cardinality* as they do not change the purpose and are typically visualized similarly. Consequently, CC Viz consists of five distinct visualizations, from which each is suited to one task from taxonomy [5]. The tool allows users to switch between visualizations and contains interactive features to view problems in greater detail.

In the remainder of this paper, we introduce the CC Viz tool with all its functionalities in section 2, after which we illustrate its maturity in section 3 and conclude in section 4.

[1]Note that guideline violations most commonly refer to deviations from a process model [5].

2. The CC Viz Tool

At https://github.com/michaelgrohs/ccviz, the CC Viz tool can be accessed. In this repository, the user can find the source code, instructions on how to run the tool, and further documentation. A demo video is available at https://github.com/michaelgrohs/ccviz/blob/master/demonstration.mov. Alternatively, it is also possible to access the tool in a hosted setting without requiring any local setup via https://ccviz-frontend.vercel.app. This relies on a hosted backend service Render (https://render.com) and a frontend service Vercel (https://vercel.com). Note that performance is significantly slower than the local setup and the backend service is sleeping per default. Thus, the functionalities are not available directly and has to be woken up. Also, the service restricts RAM to 512 MB, meaning that it cannot handle larger files (the default dataset works).

2.1. Functional Components

As illustrated in Fig. 1, a process model and an event log are required as input for CC Viz. Then, all relevant computations are performed simultaneously for all tasks once, meaning that no waiting times are required after that. This computation uses current state-of-the-art alignment-based conformance checking as basis which provides detailed feedback on skipped and inserted activities per trace as well as fitness levels. Following that, the user enters the task-specific visualizations, starting with task (1). It is possible to switch between the visualizations, moving to the next up- or downstream task in the sequence. In the following, we present the five visualizations in more detail using the travel reimbursement process from the BPI Challenge 2020 (BPIC20) [2] as running example. The corresponding process model is obtained from [6].

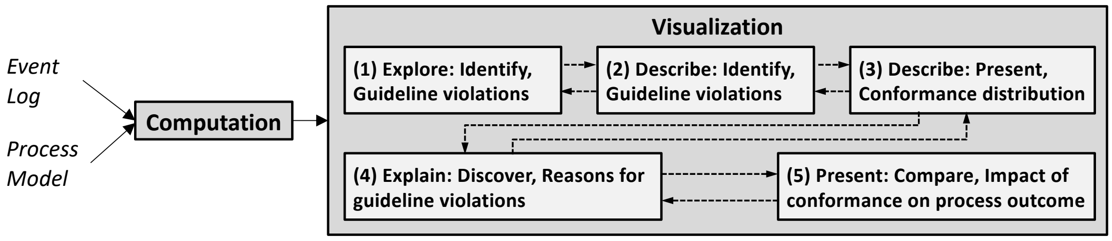
Event Log
Process Model
Computation
Visualization
(1) Explore: Identify, Guideline violations
(2) Describe: Identify, Guideline violations
(3) Describe: Present, Conformance distribution
(4) Explain: Discover, Reasons for guideline violations
(5) Present: Compare, Impact of conformance on process outcome

Figure 1: Overview of CC Viz's functional components

(1) Explore: Identify, Guideline violations. This first task aims to identify where exactly traces differ from the desired behavior defined by the model. Corresponding violations are identified by users in an exploratory manner. To address this task, CC Viz shows the provided BPMN and color codes the degree of conformance per activity, light colors indicating conform activities and darker colors indicating violations (Fig. 2). This degree of conformance is defined by the proportion of traces in which an activity has been skipped or inserted, as detected by the trace alignments. The number of skips and insertions are also portrayed for activities when hovering over them. Thus, the user can explore which activities pose problems in the process. For BPIC20, most of the process is conform but administration approval appears problematic.

[2]https://doi.org/10.4121/uuid:52fb97d4-4588-43c9-9d04-3604d4613b51

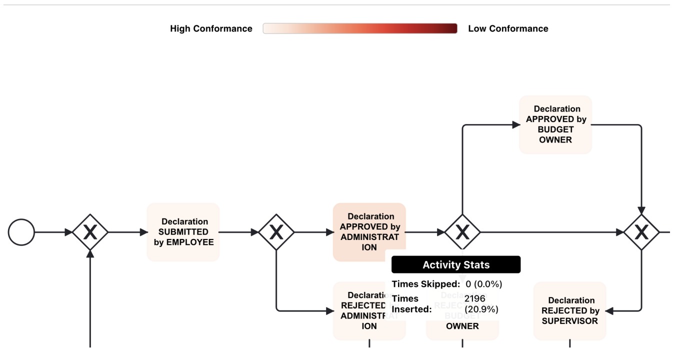
High Conformance
Low Conformance
Declaration SUBMITTED by EMPLOYEE
Declaration APPROVED by ADMINISTRATION
Declaration APPROVED by BUDGET OWNER
Declaration REJECTED by ADMINISTRATION
Declaration REJECTED by SUPERVISOR
OWNER
Activity Stats
Times Skipped: 0 (0.0%)
Times Inserted: 2196 (20.9%)

Figure 2: Visualization for Task (1)

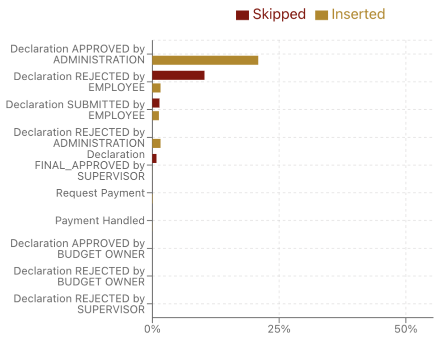

Figure 3: Visualization for Task (2)

(2) Describe: Identify, Guideline violations. The second task aims to identify what behavior was executed instead of the behavior captured in the process model. The task is similar to task (1) but does not explore the violations and rather directly describes them. For this, CC Viz contains a bar chart per activity with one bar showing skip frequency and another bar showing insertion frequency (Fig. 3). In BPIC20, administration approval is often inserted.

(3) Describe: Present, Conformance distribution. The third task describes which percentage of traces in the log fall into which conformance category. These categories are defined by trace fitness values. CC Viz shows the fitness distribution of traces with a bar chart (Fig. 4). Thereby, traces are sorted into bins with size 0.1. When clicking on a bar, the event sequences within this bar are shown. In BPIC20, most traces are conform but 134 traces deviate significantly.

(4) Explain: Discover, Reasons for guideline violations. The fourth task aims to explain which attributes are responsible for guideline violations. To address this task, CC Viz contains a chart that shows the fitness grouped by all trace and event attributes in the log (Fig. 5). Attributes are selected by the user. For categorical attributes, a bar chart is presented. For numerical attributes, a scatterplot is displayed. This allows to deduce which values might cause guideline violations. For instance, the role MISSING correlates significantly with low fitness in BPIC20.

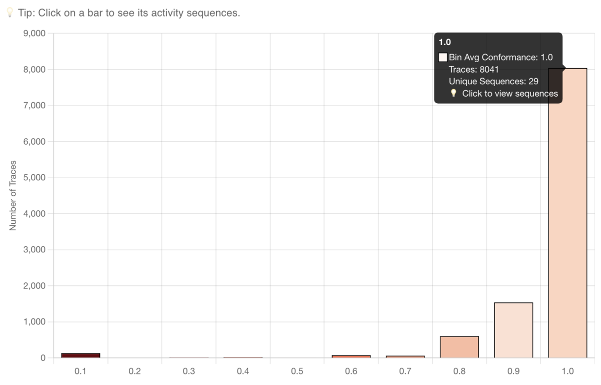

Figure 4: Visualization for Task (3)

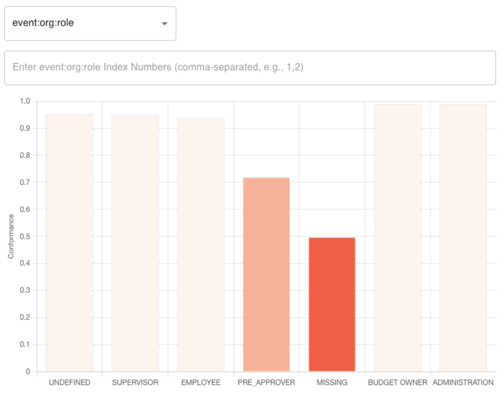

Figure 5: Visualization for Task (4)

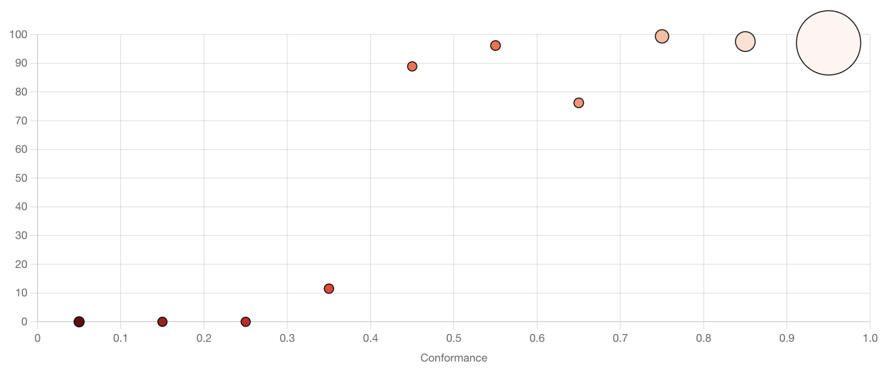

Figure 6: Visualization for Task (5)

(5) Present: Compare, Impact of conformance on process outcome. The last task aims to compare the impact of conformance on the process outcome. The process outcome is thereby defined as a binary feature, either positive (1) or negative (0). The definition of the process outcome is done by the user. Concretely, it can be defined based on the trace containing or ending with a certain key activity. Per default, it is defined as a correct execution of the last activity. For example, for BPIC20, a trace is considered to have a positive outcome if it ends with *Payment Handled*. CC Viz displays a bubble chart with the percentage of traces with a positive outcome per bin of trace-level fitness (Fig. 6). Thus, users can see if traces with lower fitness also tend to have fewer positive outcomes, which is the case for BPIC20.

2.2. Tool Architecture

In CC Viz, a Python-based back-end and a React-based front-end communicate through request and response mechanisms. The back-end utilizes Flask to ensure communication even if the front-end is closed. Further, PM4Py's trace alignments [7] handle the conformance check.

3. Maturity

We used the tool for the processes of the BPI Challenges 2012 (sub-process with A_ and O_ activities only; *12A & 12O*) and 2020 (Domestic Declarations (Dom.), International Declarations (Int.), Request for Payment (RfP), and Prepaid Travel Costs (Prep.)) obtained from [6] as well as the sepsis process [8] and the road traffic fines process obtained from [5]. These processes represent a variety of examples with respect to size, complexity, and count of recorded attributes.

Tab. 1 shows descriptive statistics for all used event log-process model pairs and required computation times. We see that CC Viz is heavily impacted by the computation of alignments, which is time-intensive for processes with many, long variants and complex models like Sepsis with 354 seconds (about 5.8 minutes). Road traffic also takes relatively long due to its size with 122 seconds (about 2.0 minutes). All other processes take less than 30 seconds. This indicates that the attribute processing for task (4) does not affect times. In summary, computations require time, mainly due to trace alignments, but remain below 6 minutes even for complex processes.

Table 1
Descriptive Statistics and Computation Times in Seconds for Used Event Logs

Process	# Traces	# Variants	Avg. # of Activities per Trace	# Activities in Model	# Event Attributes	# Trace Attributes	Computation Time
12_A	13,087	17	4.65	10	5	3	4.5 s
12_O	5,015	168	6.23	10	5	3	3.1 s
Dom.	10,500	99	5.37	10	5	5	12.4 s
Int.	6,449	753	11.19	27	5	18	24.6 s
Prep.	2,099	202	8.69	24	5	17	10.6 s
RfP	6,886	89	5.34	13	5	9	9.7 s
Road Traffic	150,370	231	3.73	13	15	1	121.6 s
Sepsis	1,050	846	14.49	22	31	1	353.8 s

4. Conclusion

We presented CC Viz, a tool for the visualization of conformance checking results. It aims to provide visualizations that are tailored exactly to the purpose that users have when applying conformance checking. To achieve that, the visualizations are based on tasks identified in a corresponding taxonomy [5] and accessible in a common sequential analysis pipeline. In the future, we aim to empirically assess whether users can use CC Viz to solve the underlying tasks.

Declaration on Generative AI

The authors used ChatGPT, DeepL for: Grammar & spelling check, Paraphrase & reword. The authors reviewed and edited the content as needed and take full responsibility for all content.

References

[1] J. Carmona, B. van Dongen, M. Weidlich, Conformance checking: foundations, milestones and challenges, in: Process mining handbook, Springer, 2022, pp. 155–190.

[2] S. Dunzer, M. Stierle, M. Matzner, S. Baier, Conformance checking: a state-of-the-art literature review, in: S-BPMONE, ACM, 2019, pp. 1–10.

[3] M. Grohs, H. van der Aa, J.-R. Rehse, Beyond log and model moves in conformance checking: Discovering process-level deviation patterns, in: BPM, Springer, 2024, pp. 381–399.

[4] L. Reinkemeyer, Process mining in action, Springer, 2020.

[5] J.-R. Rehse, M. Grohs, F. Klessascheck, L.-M. Klein, T. von Landesberger, L. Pufahl, A task taxonomy for conformance checking, Information Systems 136 (2026) 102605.

[6] M. Grohs, P. Pfeiffer, J.-R. Rehse, Proactive conformance checking: An approach for predicting deviations in business processes, Information Systems 127 (2025) 102461.

[7] A. Berti, S. J. van Zelst, W. van der Aalst, Process mining for python (pm4py): Bridging the gap between process-and data science, ICPM Demos (2019).

[8] F. Mannhardt, D. Blinde, Analyzing the trajectories of patients with sepsis using process mining, in: RADAR+ EMISA 2017, 2017, pp. 72–80.