CauseCheck: A Tool for Simulating Deviations in Event Logs with Known Root Causes

Frederik Hake[1,*], Simon Schneider[1,*], Nikolaos Theofanopoulos[1,*], Camila Gonzalez[1,*], Poey Sie Chuah[1,*], Michael Grohs[1,*] and Jana-Rebecca Rehse[1,*]

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

Abstract

Conformance checking compares process executions in an event log to a process model to detect where and how the executions deviate from the model. However, these techniques are not able to explain why deviations occur, i.e., what the root causes of deviations are. In this vein, root cause analysis techniques have been proposed, but their suitability for conformance deviations is uncertain due to a lack of appropriate evaluation data. To address this gap, this paper presents CauseCheck, a tool that simulates event logs with deviations from a process model for which the causes are known. In particular, the user defines such deviations and assigns corresponding root causes in the form of trace and event attributes. Thus, the logs can be used to show the ability to re-discover the known root causes for deviations.

Keywords

Process Mining, Conformance Checking, Root Cause Analysis, Event Log Generation

Metadata description	Value
Tool name	CauseCheck
Current version	1.0
Legal code license	Apache 2.0
Languages, tools, and services used	React.tsx, Python, PM4Py, GraphViz
Supported operating environment	Microsoft Windows
Download/Demo URL	https://github.com/FrederikHake/CauseCheck
Documentation URL	https://github.com/FrederikHake/CauseCheck/blob/main/src/frontend/public/Manual.pdf
Source code repository	https://github.com/FrederikHake/CauseCheck
Screencast video	https://github.com/FrederikHake/CauseCheck/blob/main/Demo%20CauseCheck.mp4

1. Introduction

Conformance checking aims to analyze the relation between the intended behavior of a process, captured in a process model, and the observed behavior of a process, captured in an event

ICPM 2024 Tool Demonstration Track, October 14-18, 2024, Kongens Lyngby, Denmark

*Corresponding author.

frederik.hake@students.uni-mannheim.de (F. Hake); simon.schneider@students.uni-mannheim.de (S. Schneider); nikolaos.theofanopoulos@students.uni-mannheim.de (N. Theofanopoulos); camila.gonzalez.de.aranda@students.uni-mannheim.de (C. Gonzalez); poey.chuah@students.uni-mannheim.de (P. S. Chuah); michael.grohs@uni-mannheim.de (M. Grohs); rehse@uni-mannheim.de (J. Rehse)

log [1]. Over the last years, multiple conformance checking techniques have been developed, such as rule checking, token-based replay, and alignments [2]. As output, the techniques often quantify the degree of conformance, so-called fitness. Some also provide more detailed insights. For example, alignments identify which events have been inserted or skipped [1].

Existing conformance checking techniques are able to identify how and where process executions deviate, but they are not providing any insights why a deviation occurred [2]. Having these insights into why such deviations occur could help process managers to prevent deviations. Consequently, it would be desirable to support managers by deriving causes for the deviations directly from event data, i.e., derive which attributes causally increase the likelihood of deviations [3]. Although some techniques aim to unravel root causes for problems (e.g., [3, 4]), they have not been shown to detect such causes for conformance deviations.

To assess and compare the quality of approaches that derive root causes for deviations, appropriate evaluation data is required in the form of event logs that contain deviations from a process model for which the root causes are available as a *ground truth*. This is not the case for publicly available real-life event logs, for which such a ground truth of root causes for deviations does not exist. That is a common problem when evaluating root cause analysis techniques, not only when analysing causes of deviations. Thus, evaluations are often based on illustrations on these real-life logs rather than comparisons of techniques' capabilities to a ground truth [3, 4]. The lack of appropriate evaluation data can be encountered by using simulated data with ground truth, which has not been done for conformance deviations.

To address this gap, we present the CauseCheck tool for simulating deviations from a process model in event logs for which the root cause is known. Given a process model as input that captures the intended process behavior, the tool simulates an event log and synthetically injects different types of deviations that can occur in a process. In particular, it allows users to use deviation types based on five patterns commonly used to characterize deviations [5]: *inserted*, *skipped*, *repeated*, *replaced*, and *swapped* activities (or sequences thereof). These deviations are assigned root causes in the form of trace and event attributes. For that, the user first defines these trace and event attributes. Then, whenever a particular attribute has a particular value, a deviation occurs with a user-defined likelihood. For example, one potential cause-deviation pair could be "whenever the bank is equal to Bank A, activity Z is skipped in 50% of the traces although it is required according to the model". Further, users can define noise levels, i.e., random occurrences of deviations that are not attributed to a root cause. The tool returns an event log that contains deviations and the root causes within the trace and event attributes.

2. The CauseCheck Tool

At https://github.com/FrederikHake/CauseCheck, the CauseCheck 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/FrederikHake/CauseCheck/blob/main/Demo%20CauseCheck.mp4.

2.1. Functional Components

As illustrated in Fig. 1, a process model is required as input to set up an evaluation experiment using CauseCheck. This process model describes the to-be behavior and a playout of it is synthetically changed to contain the deviation and causes. Then, the user defines general characteristics of the desired event log as well as decision point probabilities within the process models. After that, all event and trace attributes should be created. Subsequently, the user defines which deviations from the process model occur and by which trace and event attributes they are caused. Finally, the user has the option to include a level of noise. As output, the tool generates an XES file with the simulated deviations and causes. In the following, we present the steps of the tool in more detail using a loan application process as running example.

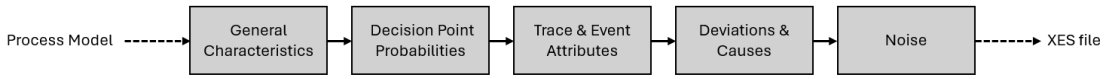

Figure 1: Overview of CauseCheck's functional components

General Characteristics. After providing a process model which captures the intended behavior in a BPMN or PNML format as input, the user is requested to specify the time-frame of the event log as illustrated in Fig. 2. Further, the size of the event log should be defined.

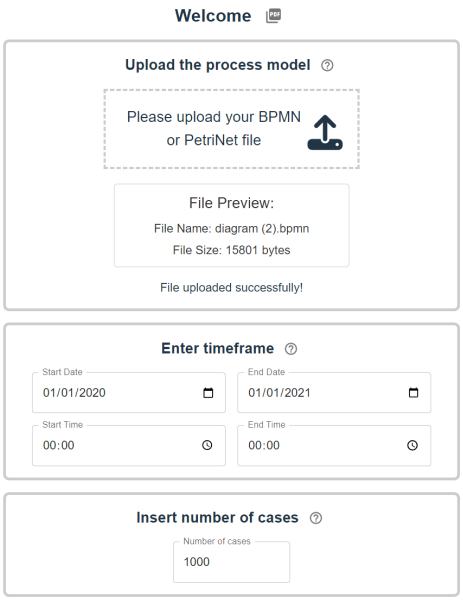

Figure 2: General Characteristics

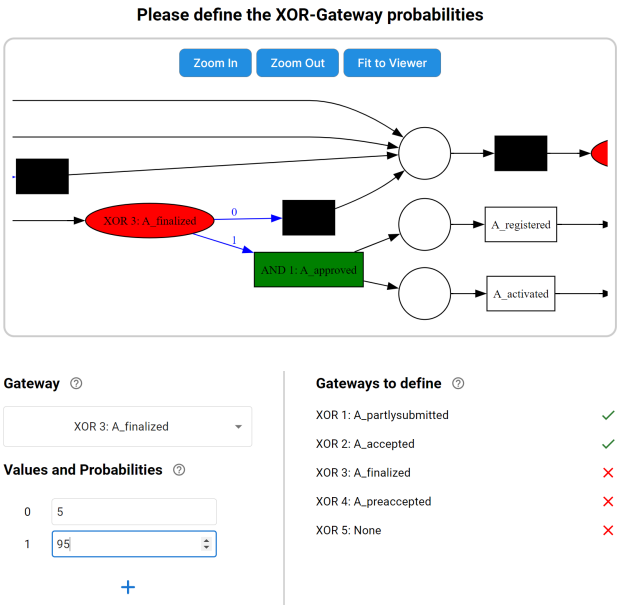

Figure 3: Decision Point Probabilities

Decision Point Probabilities. Process models often include decision points where only one of the multiple paths should be followed. There are processes where certain paths are less common than others, e.g., a cancellation of a loan application is less common than its eventual payout. To account for these cases and define the intended behavior in detail, CauseCheck requires the

user to assign decision point probabilities to all XOR-choices in the process model. As shown in Fig. 3, the tool displays the process model as a Petri net of the process in the upper part of the screen to identify decision points. Then, the user should define probabilities for each path. Per default, the simulation assumes an equal likelihood of all possible paths.

Trace & Event Attributes. The third page of the application prompts the user to define trace and event attributes which can be selected later as root causes for deviations, starting with the event attributes. For that, the user can select attributes of the types numerical, categorical, and time-related from a drop-down menu. For all event attributes, the user defines which attribute values are possible and how likely each value is for each activity the event is associated with. Thereby, only the timestamp is mandatory to be selected so that the user is able to proceed to the next page of the application. After the successful definition of all event attributes, the tool shows a summary of them. Similarly to the event attributes, the user is then prompted to define trace attributes. This particular page is optional. Again, the user may select a trace attribute type from the drop-down, define all possible attribute values, and their corresponding likelihood. For instance, Fig. 4 shows the trace attribute "Bank" with the three possible values "Bank A", "Bank B", and "Bank C".

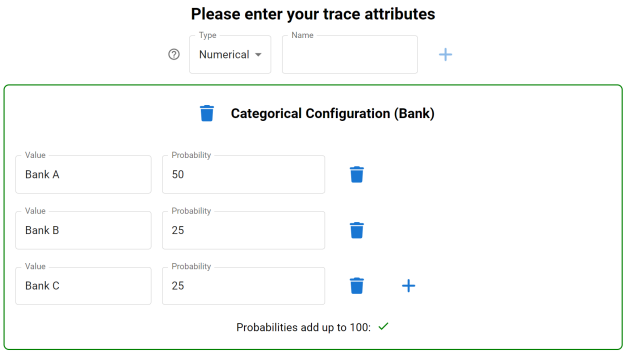

Figure 4: Trace & Event Attributes

Deviations & Causes. In the next step, the user defines the deviations as well as corresponding ground truth of causes. For that, users can select out of five commonly used deviation types [5]:

(1) *inserted*: an activity (or a sequence) is executed in addition to the intended model behavior at a point in the trace, which is defined by the user
(2) *skipped*: an activity (or a sequence) is not executed although required in the model
(3) *repeated*: an activity (or a sequence) is wrongly re-executed after it has been previously executed in accordance with the model
(4) *replaced*: an activity (or a sequence) is executed instead of another activity (or a sequence)
(5) *swapped*: two activities (or sequences) are performed in the wrong order

For each deviation, the user enters a unique identifier and also which activity (or sequence) it should refer to. For example, the user can specify that activity "A_partly submitted" is skipped. Then, a cause (or multiple causes) of the deviation should be defined. For that, users can select all trace and event attributes and choose a particular value as the cause. Further, a likelihood of deviation occurrence is required. For example, as illustrated in Fig. 5, the user can specify that "A_partly submitted" is skipped with a likelihood of 50% whenever the trace attribute "Bank" is equal to "Bank A". This likelihood corresponds to the causal effect of the attribute value "Bank A" on the skip. After specifying these details, the user adds the new deviation. This can be done for any number of deviations.

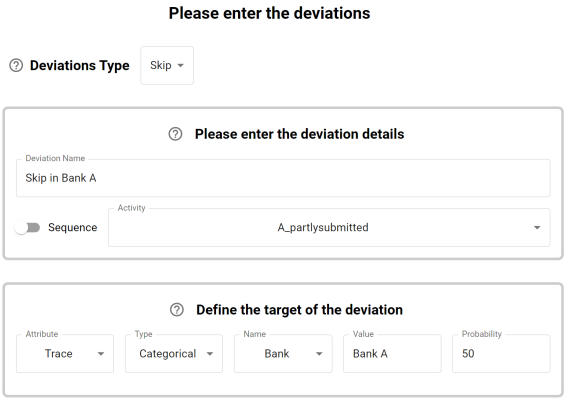

Figure 5: Deviations & Causes

Noise. In the final step, the user can include noise into the event log. This noise is defined as random occurrences of deviations with no associated cause. The user can add general noise (i.e., random occurrence of an undefined deviation), type specific noise (i.e., random occurrence of a deviation type like *skipped*) and deviation specific noise (i.e., random occurrence of a previously defined deviation). For all different options, the level of noise is assigned as a probability.

The last screen of the application is responsible for downloading the simulated event log with all its deviations and causes. The user can download both the event log with the deviations they created and a deviation-free log.

2.2. Tool Architecture

The CauseCheck tool features a Python-based back-end and a React-based front-end that communicate through request and response mechanisms. The back-end utilizes Flask-session to ensure communication with the back-end even if the front-end is closed during use. Further, PM4Py[6] handles the process model and its playout. The front-end, built with TypeScript, incorporates the Material UI React component library for flexibility and easy customization.

3. Maturity

We used the tool for processes models of the BPI Challenges 2012 (sub-process with A_ activities only; *12A*) and 2020 (International Declarations; *Int.*) obtained from [7]. 12A is rather straightforward with only 10 activities whereas Int. is more complex with 34 activities and a potential loop. Based on these models, we simulated 32 logs in total, 16 logs for each 12A and Int. In particular, we inserted either 5, 10, 15, or 20 different deviations into logs with either 100, 1,000, 10,000, or 100,000 traces. Executing these 4×4 combinations led to 16 logs per model, which we uploaded to our repository. Execution times in seconds for all 32 logs in Tab. 1 indicate reasonable computational efficiency. Thereby, times are not influenced by the number of deviations and scale approximately linearly with the number of traces. For Int., the times take substantially longer due to the complex process model but are still reasonable with 2 hours.

To show the functionality of CauseCheck, consider Fig. 6. It illustrates the occurrences of a skip of the first activity in the 12A log within the simulation of 1,000 traces. This activity should occur in every trace but is synthetically skipped in 50% of the traces associated with Bank A. Since only 50% of the traces are associated with Bank A, the deviation should on average exist

Table 1
Execution Times in Seconds for 12A and Int., subdivided by No. of Deviations and No. of Traces

12A						Int.				
# Traces \ # Devs	5	10	15	20		# Traces \ # Devs	5	10	15	20
100	2	3	2	1		100	12	10	10	8
1,000	3	2	6	3		1,000	113	109	85	118
10,000	21	9	47	25		10,000	940	753	732	646
100,000	337	323	359	322		100,000	7,258	7,115	6,808	7,142

in 25% of the traces. In our simulation, the skip occurs in 233 of 1,000 traces, indicating that, after adjusting for randomness in the probabilities, the correct number of traces contains the deviation. Further, consider Fig. 7 which shows the alignment of a different deviating trace. Concretely, A_FINALIZED and A_ACCEPTED are swapped with each other, visible as a log and model move on A_FINALIZED with a synchronous move on A_ACCEPTED in between.

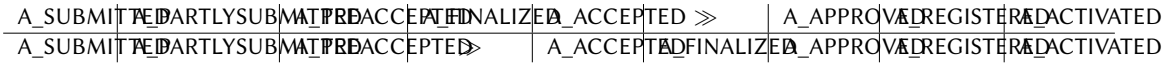

Figure 7: Alignment of Trace with Swap of A_FINALIZED and A_ACCEPTED in BPIC12

4. Conclusion

We presented CauseCheck, a tool for generating synthetic event logs with conformance deviations for which the ground truth of causes is known. It aims to provide a realistic simulation by assigning probabilities to decision points and incorporating noise. This allows researchers to evaluate tools that want to uncover root causes for conformance deviations. In particular, the output of root cause analysis techniques can be compared to the ground truth, quantifying whether the correct causes for the deviations are detected. In the future, we want to analyze the capabilities of techniques to re-discover deviation causes and potentially propose our solution for the task.

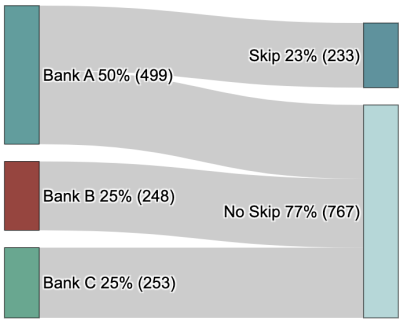

Figure 6: Distribution of Skip Based on Trace Attribute Bank

References

[1] J. Carmona, B. van Dongen, A. Solti, M. Weidlich, Conformance Checking - Relating Processes and Models, Springer, 2018.

[2] M. Grohs, J.-R. Rehse, Attribute-based conformance diagnosis: Correlating trace attributes with process conformance, in: ICPM Workshops, 2023, pp. 203–215.

[3] M. Qafari, W. Aalst, Feature recommendation for structural equation model discovery in process mining, Prog Artif Intell (2022) 1–25.

[4] Z. Bozorgi, I. Teinemaa, M. Dumas, M. La Rosa, A. Polyvyanyy, Process mining meets causal machine learning: Discovering causal rules from event logs, in: ICPM, 2020, pp. 129–136.

[5] M. Hosseinpour, M. Jans, Auditors' categorization of process deviations, Journal of Information Systems 38 (2024) 67–89.

[6] 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).

[7] M. Grohs, P. Pfeiffer, J.-R. Rehse, Business process deviation prediction: Predicting nonconforming process behavior, in: ICPM, 2023, pp. 113–120.