Trident: Generating Noisy Synthetic Processes with Ground Truth

Dominique Sommers, Durairaja V. Varadarajan and Natalia Sidorova

Eindhoven University of Technology, Mathematics and Computer Science, Eindhoven, the Netherlands

Abstract

We present Trident: a tool allowing to modify a process model in order to impute realistic behavioral noise in the modeled behavior and generate an event log that can be used to evaluate the performance of process mining techniques on realistically noisy data with knowledge of the "ground truth", i.e., the true behavior of the system. Traditional approaches for generating noisy process data take a process model that represents the true process, generate a simulated event log and impute noise in the log, while in reality, both the recorded and the modeled behavior are imprecise representations of a process, and noise in the log often follows certain patterns. In our approach, noise is introduced through a series of model transformations applying user-defined deviation patterns to a designed base process model, which creates a basis for more advanced evaluation of process mining methods and tools.

Keywords

Synthetic process data, realistic noise, patterns, model transformations

1. Introduction

Process mining is a powerful tool for understanding and improving business processes by exploring the behavior of the system recorded in an event log in order to discover the process model that could generate the observed behavior or to compare the recorded behavior to normative process models. The challenge in process mining arises from the fact that the system's true nature S is often unknown, while the recorded log L is noisy and incomplete, and the true model M is either unknown (in the context of process discovery) or imperfect (in the context of conformance checking).

Deviating behavior, or noise can be categorized into random and behavioral noise. While simple random noise arises from errors in logging or modeling and includes missing or incorrectly included events, behavioral noise entails the violation of structural patterns and dependencies within a process. It is crucial that process mining algorithms are able to work in the presence of random and behavioral noise.

One of the good practices in the field of process mining is that most of the developed methods are evaluated on data from real-life processes, which is inherently noisy. However, no ground truth is available for such data sets and the conclusions can only be made based on the feedback from the process owner.

ICPM'23: International Conference on Process Mining, October 23–27, 2023, Rome, Italy

d.sommers@tue.nl (D. Sommers); d.v.varadarajan@student.tue.nl (D. V. Varadarajan); n.sidorova@tue.nl (N. Sidorova)

Another standard practice is evaluating process mining methods in a controlled environment where the ground truth is known, i.e., having knowledge about L and M. The usual approach to creating such data is to simulate (play out) a designed process model and introduce random noise in the resulting event log through log manipulations like insertion, deletion, and swaps [1]. The process model represents the true process, and the log represents the recorded behavior with data quality issues [2]. This method does not cover the full spectrum of evaluation challenges, since it does not take into account imperfections in the modeled behavior as well as behavioral noise in M and L.

Typical deviations and behavioral noise can be defined as (anti-)patterns, e.g., the multitasking pattern. As a running example, consider a delivery process, where a deliverer should first ring a customer and then deliver a package without switching from one package to another between ringing and delivering. An example of deviating behavior is multitasking between packages: i.e., the deliverer rings another door before handing over the first package, which the normative process model does not allow.

Our tool, Trident, is designed to modify a given business model M_0 by applying model transformation patterns, with each pattern addressing a specific type of behavioral or random noise. The resulting process model M' represents the real process execution including deviating behavior, and is used to generate an event log L. During the simulation of M', random noise can additionally be applied to L through log manipulations. Complete knowledge of the "true behavior" S is retained through the model transformations applied to M_0 and the log manipulations applied to L, and both M_0 and L serve as imprecise representations of S.

2. Generating a "Noisy" Process Model

To set up an evaluation experiment using Trident, one takes a real-life process model or sketches a hypothetical process with a set of appropriate deviations. With this tool, these deviations are applied to a designed base model through a series of model transformations to simulate a realistic synthetic process including deviating behavior.

2.1. Model transformations

An overview of the usage of Trident is shown in Fig. 1, with the designed base model M_0 on the top and an (extendable) set of patterns $(\pi_1, \pi_2, \pi_3, \dots)$ on the right. A pattern π is in itself a process model which is partitioned into *match* and *create* components: π_m and π_c. Trident applies the model transformation of π on a process model M through a user-defined mapping function f, denoted by $\Psi(M, \pi, f) = M'$, as depicted in the red trident shape. f defines a mapping from π_m to M, which dictates how the elements from π_c are added to M. Through the model transformation, M' contains the created components of π_1, connected to the matched elements of M_0, via f. This process is repeated until all deviation patterns are included in M, after which the process model can be either exported or immediately simulated by playing it out from initial to final marking in the tool.

Fig. 1 illustrates an example model transformation $\Psi(M_o, \pi_1, f) = M'$, with $f(p_r) = p_d$ and $f(\bar{p}_r) = \bar{p}_d$. The components' colors in π_1 show the partitioning into matched (blue) and created (green). π_1 models the behavioral deviation of multitasking of a resource, by allowing a resource

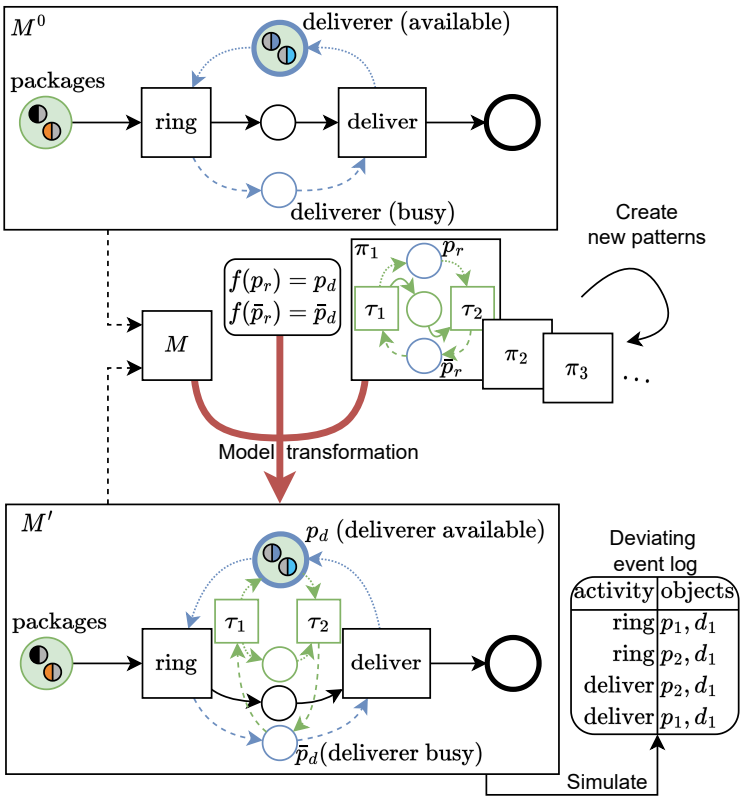

Figure 1: Overview of Trident with running example.

to switch from the state "busy" to "available" at any point in time. By applying π_1 to M_0 on the deliverer resource, the scenario described in Sec. 1 is enabled where a deliverer can ring another door before handing over the first package.

The tool operates on a generalized ν-net formalism, which is a restricted version of colored Petri nets and encompasses Petri net extensions like resource-constrained (RC) ν-nets [3], typed Petri nets with identifiers (t-PNIDs), typed Jackson nets [4], as well as Object-Centric Petri nets [5]. Such a ν-net acts as the base process model for the tool and as M for the methods using the data. The user interface is focused towards resource-constrained RC ν-nets, distinguishing between the place types being regular, resource available, and resource busy. Our running example is modeled as an RC ν-net as shown in Fig. 1. In each iteration, after the selection of a π and f, the model transformation to be applied is validated to ensure that the resulting process model retains its soundness.

2.2. Creating new deviation patterns

There is a provided list of possible deviations that can be applied to the base process model. This includes multitasking from Fig. 1, as well as temporarily increasing capacities, neglecting resources in activity executions, overtaking in first-in-first-out queues, resources switching

roles, and more. This list is easily extendable, where one can model a new deviation as a Petri net from an explanation of deviating behavior, e.g., skipping an assumed to be necessary activity like *deliver* in M^0. Designing new deviation patterns is supported in the tool by providing the user with a template and instructions on how to create the pattern. The example mentioned is considered behavioral noise where a modeling pattern is violated in the true behavior of the system. Random noise can be trivially modeled similarly by deviation patterns. Skipping activities is an example that is included in the list as well.

2.3. Simulation

After iteratively applying model transformations to the base process model, M includes all deviations deemed realistic by the user. The simulation module is a play out of M from the provided initial to final marking, with the option to set a limit of the number of transition firings, in case of infinite behavior. Probabilities are modeled through sampling a waiting time for transition firings from the moment they are enabled. In case the transition is not enabled at the scheduled time anymore, it is canceled. The simulation is basic in terms of probabilities of which transitions to fire, i.e., it does not take into account any other dependencies than the sampled scheduling time from the moment it is enabled. If one requires a more advanced simulation, process model M can be exported to `.pnml` to be used in other tools.

3. Availability and Maturity

The tool is based on Python and can be operated through either a Flask GUI, a command line interface, and/or run from other Python code, to generate ground truth synthetic process data. The source code, an installation manual, and a screencast are available at gitlab.com/vignesh_dv/mira/-/tree/paper/mira/pattern.

We have used the tool throughout our research project CERTIF-AI involving various industry partners, for which we model hypothetical assembly processes fitting the companies' data including the behavior of operators as resources. We add potential violations on inter-case dependencies via these resources to generate a true representation of reality with realistic and explainable deviations. With this synthetic process, we can evaluate our methods which aim to reveal the true nature of S from the imprecise representations M and L.

4. Conclusion

We developed an open-source tool for generating a synthetic process with realistic noise that is simply random as well as behavioral, where the ground truth S is known together with imprecise representations of S in the form of the process model M_0 and the simulated event log L. This allows for evaluation of process mining methods where both L and M_0 are analyzed to reveal information about S, like in conformance checking, log repair, model repair, and performance analysis. Unlike traditional approaches, where a process model M_0 denotes the perfect representation of S and only the generated event log L contains noise, Trident takes an available business model M_0 and constructs a process model M' that can serve

as the representation of the real process execution, using behavioral deviation patterns (e.g., multitasking or redo), or be used for the generation of a noisy event log, using log noise patterns (e.g., delayed logging for certain event types). Complete knowledge of S is retained through the transformed model M' and the simulation method.

The tool supports generalized ν-nets, making it applicable for many Petri net extensions, however, the GUI is currently focused towards only resource-constrained ν-nets. We aim to expand this to generalized ν-nets in the future.

Acknowledgments

This work is done within the project "Certification of production process quality through Artificial Intelligence (CERTIF-AI)", funded by NWO (project number: 17998).

References

[1] T. Jouck, B. Depaire, Generating artificial data for empirical analysis of control-flow discovery algorithms: A process tree and log generator, Business & Information Systems Engineering 61 (2019) 695–712.

[2] R. J. C. Bose, R. S. Mans, W. M. v. Aalst, Wanna improve process mining results?, in: 2013 IEEE symposium on computational intelligence and data mining (CIDM), IEEE, 2013, pp. 127–134.

[3] D. Sommers, N. Sidorova, B. F. v. Dongen, Aligning event logs to Resource-Constrained Petri nets, in: International Conference on Applications and Theory of Petri Nets and Concurrency, Springer, 2022, pp. 325–345.

[4] J. M. E. van der Werf, A. Rivkin, A. Polyvyanyy, M. Montali, Data and process resonance: Identifier soundness for models of information systems, in: International Conference on Applications and Theory of Petri Nets and Concurrency, Springer, 2022, pp. 369–392.

[5] W. M. v. Aalst, A. Berti, Discovering object-centric Petri nets, Fundamenta informaticae 175 (2020) 1–40.