Adaptive Process Log Generation and Analysis with Next(Log) and ML.Log

Cartwright, Dyllan; Sterie, Radu Andrei; Yadegari Ghahderijani, Arash; Karastoyanova, Dimka

Published in:
Enterprise Design, Operations, and Computing. EDOC 2023 Workshops - IDAMS, iRESEARCH, MIDas4CS, SoEA4EE, EDOC Forum, Demonstrations Track and Doctoral Consortium, 2023, Revised Selected Papers

DOI:
10.1007/978-3-031-54712-6_21

Adaptive Process Log Generation and Analysis with *Next(Log)* and *ML.Log*

Dyllan Cartwright(✉), Radu Andrei Sterie(✉), Arash Yadegari Ghahderijani(✉), and Dimka Karastoyanova(✉)

Information Systems Group, University of Groningen, Groningen, The Netherlands
{a.yadegari.ghahderijani,d.karastoyanova}@rug.nl

Abstract. In this paper we present a tool for adaptive process log generation and analysis of the correlation between KPI (Key Performance Indicator) values and changes in adaptive processes. The tool features a component called *Next(Log)* helping users to generate initial business process logs using any preferred method and subsequently allows them to adapt these logs based on their own defined rules while ensuring an intuitive and coherent user interface. The adapted logs are then used for log analysis with the *ML.Log* component, which employs machine learning techniques to find patterns of matching KPI values and adaptation injections in the logs. The tool therefore supports the research on the challenges imposed by the lack of sufficient amount of data from adaptive process logs and the open issues in identifying at what KPIs values changes are required and what kind of changes would have the best impact on the process performance at run time.

Keywords: Runtime process adaptation · Adaptive process log generation · KPI-to-adaptation correlation · Synthetic process event logs

1 Introduction

Runtime process adaptation is one of the ways to enable autonomous process performance improvement [3] for augmented processes [2] enacted by process-aware information systems. Significant research results of the fields of process mining, predictive and prescriptive process monitoring [6] have established the fundamentals of learning from available process event logs for discovering processes and identifying potential KPI violations in process behaviour. One of the significant challenges this kind of research faces is the lack of available process event logs, and in particular such that containing known adaptations of any of the process dimensions - control and data flow, activity implementations and changes in (human) resource availability. The other challenge the community is facing is the matching of the reasons for changes made in processes to concrete adaptation actions. *If we can learn what reasons for changes have been and their impact on the process performance, only then we can recommend (automatically*

T. P. Sales et al. (Eds.): EDOC 2023 Workshops, LNBIP 498, pp. 331–337, 2024.
https://doi.org/10.1007/978-3-031-54712-6_21

or not) a specific adaptation action to be enacted into the affected running process instance.

In this paper we present a tool that has two main objectives: 1) to generate synthetic event logs of adaptive processes to alleviate the impact of the challenge mentioned above and 2) to analyse the adapted logs to identify correlations between the adaptations made and the KPIs values for which an adaptation was needed. The process logs it generates, using its *Next(Log)* component, are based on existing process event logs and are subsequently extended with control flow adaptation events following rules that users can define using the tool. The analysis of the extended logs the tool can perform, using its *ML.Log* component, is based on several well-known ML (Machine Learning) algorithms and identify the value of the KPIs that present a reason for an adaptation.

Our tool allows for user friendly adaptive log generation by easy import of original process logs and intuitive adaptation rule definitions based on formal grammar implemented in a rule editor. Furthermore, the tool automates the adaptive process log analysis pipeline and its architecture allows for future extensions of the algorithms used for that purpose.

We present our tool along the following paper structure: The tool's architecture and implementation are presented in Sect. 2. We show an example of adaptive process log generation and analysis in Sect. 3 and in Sect. 4 we discuss the limitations of the current version of the prototype. We conclude the paper and point to future improvements in Sect. 5.

2 Tool Components and Implementation

The key components of the tool[1] (see Fig. 1) are the *Next(Log)* component for generating adaptations in process logs based on predefined rules and the log analysis tool *ML.Log* that can use adaptive process logs to discover correlation between adaptations on the one hand and the reasons for the adaptations, like specific values of KPIs, on the other.

Next(Log) is a flexible tool designed to generate and adapt business process logs[2]. Its key component is a 'Rules Editor' page, which allows users to define custom rules for log adaptation. The tool provides a user-friendly interface, making it intuitive to define these adaption rules. Users can create rules that apply to specific traces and are built upon the trace's KPI metrics, for example, duration and cost.

Next(Log) was written in Python and used PySide6 for creating the user interface, a Python binding for the Qt framework. The PLY (Python Lex-Yacc) library was used to create a simple parser for the user-defined rules. Finally, the xml library in Python was utilised to interpret both the MXML and BPMN files, which are currently the only allowed input formats for process models and process event logs.

[1] The tool is available in https://github.com/aryadegari/Next-ML-Log/.

[2] The initial logs will need to be provided in MXML format. *Next(Log)* has only been tested on synthetic logs generated by the BIMP simulator.

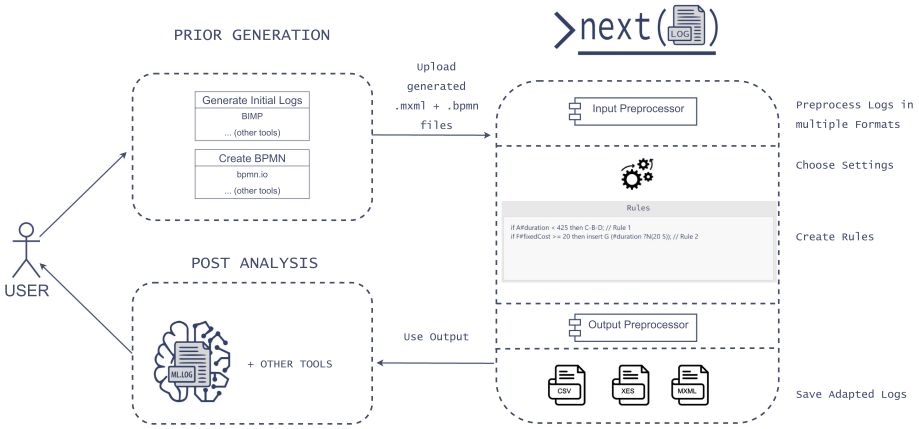

Fig. 1. Architecture of the Tool.

ML.Log is an application for using machine learning methods on adaptive business process logs. It automated the data processing pipeline and thus enables the user to easily use different machine learning models, find the best parameters, save the models for future use and test saved models on different logs.

ML.Log was also written in Python with PySide6. The software utilises the popular Python library Scikit-learn to build various machine learning models, including Decision Trees, Random Forest Classifiers, and K-Nearest Neighbours (KNN). Additionally, the sklearn-glvq module was employed to develop GLVQ models. For visualisation, Graphviz and Matplotlib were used.

3 Example of Generating and Analysing an Adaptive Process Log

In the following section, we will demonstrate a very straightforward example of adapting business process logs using *Next(Log)*, followed by *ML.Log*'s analysis of the adapted logs. A demonstration video is available[3].

Log Adaption with *Next(Log)*

Suppose we had the process model shown in Fig. 2, with the following user-defined rule[4] focusing on time as KPI:

```
if C#duration > 575 then insert K (#duration ?N(100 15));
```

The above rule can be interpreted as follows: For each trace in the initial logs, if the duration of event C was greater than 575 s, then introduce a new event K immediately after C. The duration of event K is sampled from a normal distribution with $\mu = 100$ s and $\sigma = 15$ s.

[3] https://doi.org/10.6084/m9.figshare.24082083.v1.

[4] To see all possible rules that can be made and their notation/formal grammar, please refer to [1].

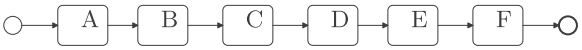

Fig. 2. Sample process model (`X#duration` $\sim N(500, 50)$ $\forall$ X $\in$ {A,..,F})

In Tables 1 and 2 we present the original process log and the adapted log using the adaptation rule from above. The presented results have undergone filtering, resulting in the exclusion of numerous traces and columns. This selection aims to try highlight elements for demonstration purposes.

Table 1. Original Logs

trace_id	C#duration	D#start_time	K#duration	_End#start_time	_End@duration	_Path
184	520.272	2023-07-19T22:02:36.535	0	2023-07-22T22:51:24.642	2877.945	"_Start,A,B,C,D,E,F,_End"
27	576.532	2023-07-18T08:37:50.625	0	2023-07-19T17:25:32.212	2981.662	"_Start,A,B,C,D,E,F,_End"
30	478.811	2023-07-18T09:52:33.449	0	2023-07-19T19:59:50.737	3013.945	"_Start,A,B,C,D,E,F,_End"
167	603.946	2023-07-19T18:43:35.746	0	2023-07-22T20:14:34.943	3018.123	"_Start,A,B,C,D,E,F,_End"

Table 2. Adapted Logs

trace_id	C#duration	D#start_time	K#duration	_End#start_time	_End@duration	_Path	_Label
184	520.272	2023-07-19T22:02:36.535	0	2023-07-22T22:51:24.642	2877.945	"_Start,A,B,C,D,E,F,_End"	0
27	576.532	2023-07-18T08:39:32.704908	102.079	2023-07-19T17:27:14.291908	3083.741908	"_Start,A,B,C,K,D,E,F,_End"	1
30	478.811	2023-07-18T09:52:33.449	0	2023-07-19T19:59:50.737	3013.945	"_Start,A,B,C,D,E,F,_End"	0
167	603.946	2023-07-19T18:45:20.822520	105.076	2023-07-22T20:16:20.019520	3123.19952	"_Start,A,B,C,K,D,E,F,_End"	1

Traces 27 and 167 trigger the specified rule. The "_Path" column updates correctly, indicating the adapted sequence of events. The "_Label" column is also updated to indicate whether an adaptation occurred or not. Additionally, the insertion of event K is observed, with its `duration` randomly sampled from $N(100, 15)$. Furthermore, `_End#start_time` and `_End@duration` are updated appropriately in response to the adaptation.

Adaptive Process Log Analysis with *ML.Log*

After adapting the logs shown in Table 2, we conducted an analysis using the machine learning techniques implemented by *ML.Log*. The results of this analysis are presented below (see Table 3).

Figure 3 represents the decision-making rule that has been retrieved by the Random Forest model as the best performing one and as visualized by the tool, while in Fig. 4 we show the results of the KNN algorithm.

For a detailed analysis of the results, you may refer to [5], which evaluates the performance of all four models using different metrics such as F1 score, recall, and accuracy, providing a quantitative measure of their performance with different logs.

Table 3. Classification Reports for Given Logs.

Model		precision	recall	f1-score	support
Random Forest	class 0	1.00	1.00	1.00	276
	class 1	1.00	1.00	1.00	24
	macro avg	1.00	1.00	1.00	300
	weighted avg	1.00	1.00	1.00	300
Decision Tree	class 0	1.00	0.99	0.99	276
	class 1	0.92	0.96	0.94	24
	macro avg	0.96	0.98	0.97	300
	weighted avg	0.99	0.99	0.99	300
KNN	class 0	0.99	1.00	0.99	276
	class 1	1.00	0.83	0.91	24
	macro avg	0.99	0.92	0.95	300
	weighted avg	0.99	0.99	0.99	300
GLVQ	class 0	0.95	1.00	0.97	276
	class 1	1.00	0.33	0.50	24
	macro avg	0.97	0.67	0.74	300
	weighted avg	0.95	0.95	0.93	300

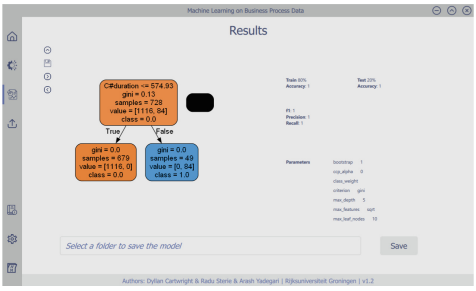

Fig. 3. Visualisation of a found Tree using a Random Forest Classifier in *ML.Log*.

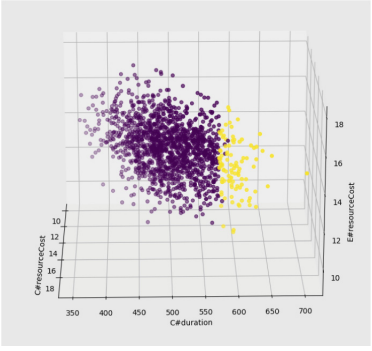

Fig. 4. Visualisation of the Labels assigned by a KNN model found within *ML.Log*.

4 Maturity

Both components of the tool have some limitations which we will discuss here.

Firstly, *Next(Log)* assumes that user-defined rules do not overlap (i.e. maximum of one rule may apply to each process case), as overlapping rules may lead to unpredictable and ambiguous outcomes. Secondly, while it is designed to work with any standard BPMN and MXML files, it has been predominantly tested

with files generated by specific tools (bpmn.io and BIMP), and minor issues may arise when using other generative tools. Moreover, the tool assumes there are no loops within the process models, and is limited to parallel and exclusive (XOR) gateways. Additionally, it does not support nesting of gateways within process models.

The current implementation of *ML.Log* includes GLVQ, KNN, Decision Trees, and Random Forest as the four implemented machine learning models, evaluated using various metrics for performance assessment. However, the limited selection of models and performance metrics may not encompass the full range of possible insights.

5 Conclusions

In this paper we presented a tool for adaptive process log generation and analysis. The current version of the tool has two components: i) *Next(Log)*: based on a BPMN process model and a corresponding process log it allows for extending the logs with different process adaptation, simulating the fact that process instances have been adapted during their execution; ii) *ML.Log* automates the data processing pipeline for analysis of the adapted process logs - currently the analysis is based on several ML algorithms.

More specifically, the objective of *Next(Log)* was to enable users to generate initial business process logs using their preferred method and subsequently adapt them based on custom rules. As demonstrated earlier and in [1], all adapted logs performed as intended. While the current version lacks the ability to create more complex adaptations, *Next(Log)* is designed so that it can be extended further. With improvements, it could serve as a valuable platform for developers and researchers to test, build and integrate with analysis tools like *ML.Log* and be a catalyst for exciting research in the field.

The objective of *ML.Log* was to find connections between adaptations and their impact on KPIs in adaptive business process logs using multiple machine learning techniques.

As the tool is currently requiring a BPMN process model as input in addition to a corresponding process log, in order to improve its usability, in future the tool can be extended with a process discovery component based on existing process discovery algorithms like PM4PY, so that the process model is discovered from the process event logs. Integration with approaches like e.g. [4] and [6] will draw upon works in prescriptive process monitoring and analytics.

References

1. Cartwright, D.: A tool for log generation of adaptive business processes. B.Sc. thesis, University of Groningen (2023). https://fse.studenttheses.ub.rug.nl/31012/
2. Dumas, M., et al.: Augmented business process management systems: a research manifesto. CoRR abs/2201.12855 (2022). https://arxiv.org/abs/2201.12855

3. Ghahderijani, A.Y., Karastoyanova, D.: Autonomic process performance improvement. In: EDOC Workshops 2021, pp. 299–307. IEEE (2021). https://doi.org/10.1109/EDOCW52865.2021.00061
4. de Leoni, M., Dees, M., Reulink, L.: Design and evaluation of a process-aware recommender system based on prescriptive analytics. In: ICPM 2020, pp. 9–16 (2020). https://doi.org/10.1109/ICPM49681.2020.00013
5. Sterie, R.A.: Adaptive business process analysis using machine learning algorithms. B.Sc. thesis, University of Groningen (2023). https://fse.studenttheses.ub.rug.nl/31141
6. Yadegari Ghahderijani, A.: Change recommendation in business processes. In: Troya, J., et al. (eds.) ICSOC 2022. LNCS, vol. 13821, pp. 334–340. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-26507-5_29