SWORD Extractor
Pulling Workarounds out the Eventlog

Wouter van der Waal*, Inge van de Weerd and Hajo A. Reijers

Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, the Netherlands

Abstract

While workarounds can be used to improve processes, discovering them may be challenging. In earlier work, we showed that the Semi-automated WORkaround Detection (SWORD) framework can be used to detect workarounds in event data. Starting from an event log, the SWORD Extractor demonstrated in this work guides the user through all the necessary steps to find cases that may indicate workarounds, without requiring extensive technical experience. The tool's design supports the implementation of additional patterns for future research.

Keywords

Workarounds, Automated detection, Event logs, Process Mining, Business process analysis

Metadata description	Value
Tool name	SWORD Extractor
Current version	1.0.0
Legal code license	MIT License
Languages, tools and services used	R, gwidgets2, tcltk
Supported operating environment	Microsoft Windows, macOS, GNU/Linux
Download/Demo URL	https://git.science.uu.nl/w.g.waal/sword/-/releases/1.0.0
Documentation URL	https://git.science.uu.nl/w.g.waal/sword
Source code repository	https://git.science.uu.nl/w.g.waal/sword
Screencast video	https://youtu.be/82-Mg81CVp0

1. Introduction

To streamline processes, many organizations define how people should work in procedures [1]. These are usually not designed to predict and describe any and all eventualities that may occur in the process. If a worker observes an obstacle to follow the procedure, such as a missing signature from a colleague who is on vacation, a system that is offline due to maintenance, or simply has too little time, they may find an alternative path to work around the block [1, 2].

Once workarounds are discovered, process managers can use them to improve the general process in various ways [3]. However, this requires the discovery of workarounds. Previous research searched for new workaround types using qualitative methods such as interviews [3, 4]

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

*Corresponding author.

w.g.vanderwaal@uu.nl (W. v. d. Waal); g.c.vandeweerd@uu.nl (I. v. d. Weerd); h.a.reijers@uu.nl (H. A. Reijers)

0000-0001-6514-7570 (W. v. d. Waal); 0000-0003-3420-6923 (I. v. d. Weerd); 0000-0001-9634-5852 (H. A. Reijers)

or used qualitative measures to recognize workaround instances of known types in event logs [5]. Building upon both methods, recent work into workaround mining has shown that new workaround types can also be discovered with a data-driven approach using the Semi-automated WORkaround Detection (SWORD) framework [6]. This framework defines various patterns that may indicate workarounds from an event log. Each pattern has individual requirements, such as accurate timestamps or actor roles, that determine if it is applicable to the event log. As the framework is modular, only a selection of the patterns can used based on the available data.

However, the practical use of the framework is not straightforward. The requirements differ strongly between the patterns and are not always common in event logs. As such, this requires programming knowledge to work with the framework even besides general implementation issues. To allow workaround mining without a technical background, we introduce the SWORD Extractor which makes the application of the SWORD framework much easier for anyone interested in discovering workarounds. Given an event log, it guides the user through selecting patterns, and applying them, removing the need for specialized skills.

In the following, we explain how the tool can be used in Section 2, discuss its maturity in Section 3, and discuss future work and conclude in Section 4.

2. Functionality

As can be seen in Figure 1, the SWORD Extractor consists of four sections: The top left section shows the *Case View*, the top right section shows the *Selection List*, the bottom left shows the *Patterns Scores*, and the bottom right shows the *Pattern Options*. In the following, we will discuss the main features of the tool per section, in order of usage. Afterward, we will discuss how the results can be used for research outside of the tool.

2.1. Loading an Event Log

The first step in using the SWORD Extractor is selecting which columns in the event log link to the columns in the SWORD framework. When run, an OS-dependent popup directly asks the user to load an event log. This log is expected to be in .csv format, where columns are separated by commas. There are no other requirements besides the file being an event log: While every line should contain one event, there is no requirement on the number of columns in the log or their exact content. In the next step, the event log columns should be linked to the SWORD format using the options in the top right *"Selection List"*.

There are six columns available: "CaseID" to distinguish between cases, "Activity" to identify the activity performed in the event, "Start time" which indicates the moment the event began, "End time" which shows when the event finished, "Log time" as the time the event was logged in the system, and "Resource" to show who performed the event. Note that "Start time" is the main time-based column and is required for all time-related patterns. Thus, even if the only available timestamp is technically the logging time, it should still be matched to "Start time" instead.

Not all columns need to be selected. Only the CaseID is strictly required to use the tool. If certain data is not available in the event log, it can simply be left empty as is the case in Figure 1

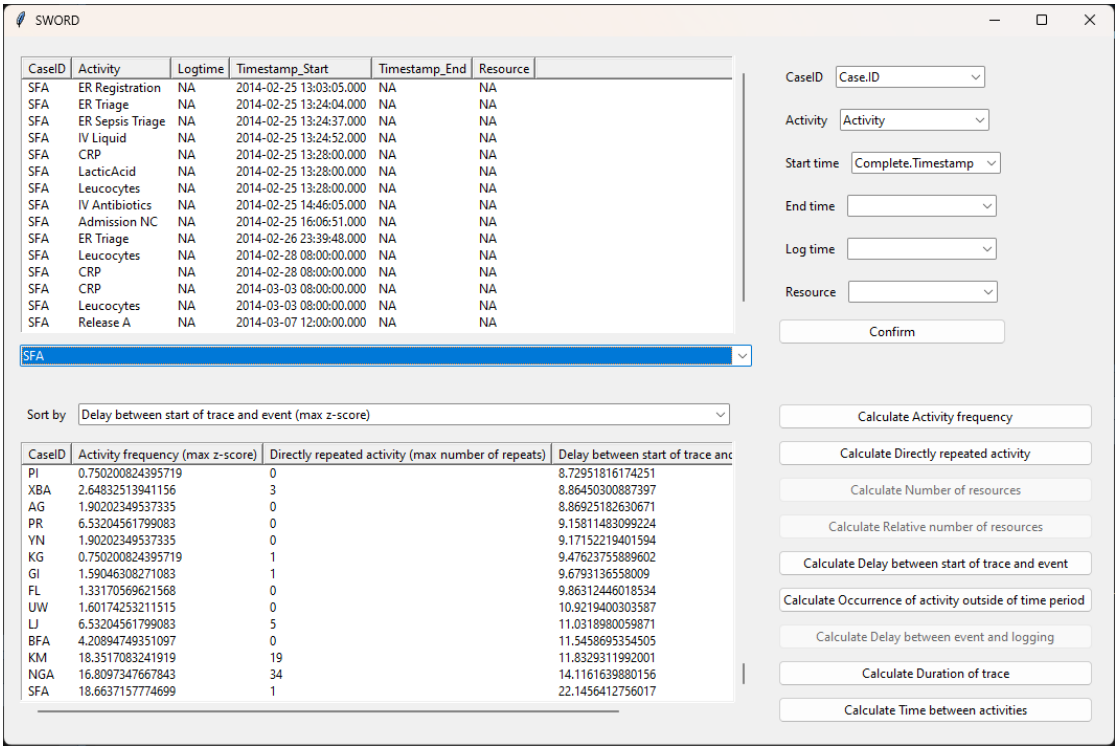

Figure 1: Screenshot of SWORD using the Sepsis event log [10]. The top left section shows the *Case View*, the top right section shows the *Selection List*, the bottom left shows the *Patterns Scores*, and the bottom right shows the *Pattern Options*

for “End time”, “Log time”, and “Resource”. However, any other column that is not matched to the event log may limit the number of SWORD patterns that can be applied at a later point.

2.2. Pattern Options

Once the right columns from the event log are loaded, all patterns that may be applied can be selected from the *Pattern Options* in the bottom right part of the interface. All patterns that have their requirements satisfied are automatically made clickable, while the remaining patterns stay grayed out. This can be seen in Figure 1. As there are no log times or resources provided, the patterns concerning the number of resources or the delay until logging cannot be calculated, while all other patterns are available.

2.3. Pattern Calculation

Any pattern available in the selection field can simply be calculated by clicking on it. Once this happens, the tool calculates the scores for this pattern per case and immediately shows the results in the *Pattern Scores* view in the bottom left. Patterns may also be re-calculated, which can be useful if the alignment to the SWORD format changes. In this case, the pattern will be

overwritten in the table.

Depending on the pattern, size of the event log, and the system used, the time it takes to calculate a pattern can differ strongly. The simpler patterns, such as “Duration of Trace” are usually finished within seconds, while more complex patterns like “Delay between start of trace and event” can take more time. Each new pattern will be shown in an additional column in this view. The patterns can be inspected in more detail by sorting with the “Sort by” drop-down menu above the table to find the traces that deviate strongest according to the specific pattern.

2.4. Case Filtering

Cases of interest, such as cases with high pattern scores, can be inspected in the top left *Case View*. This view always shows the six SWORD columns. If a column has not been matched to the event log, all values will show “NA”. The drop-down menu below the table can be used to select a specific case.

In Figure 1, case SFA is selected, as it showed the highest Z-score for the “Delay between start of trace and event” pattern. In the Case View, we can see that, according to the event log, “ER Triage” has been executed a second time more than a day after initial registration. As patients in the ER rarely stay more than 24 hours [6], this is likely an example of a delayed ER release registration workaround.

2.5. Continue with SWORD Scores

While working with the SWORD Extractor, the Pattern Scores are constantly saved to the R working directory with the name “pattern_scores”. This means that at any point in time, this table can be used in analyses in R outside of the tool. The table may also be exported using normal R-syntax for further research using different tools.

3. Maturity

While the GUI of the tool has not been used before, the SWORD framework has been applied in multiple studies to discover workarounds. Initially, the patterns in the framework have been defined in [7], based on a literature review and previous qualitative case studies in healthcare. In this study, three well-known workaround types were discovered in a real hospital dataset.

These patterns were further defined and implemented in a healthcare multiple case study where 11 previously undiscovered workaround types were found [6]. This showed that the framework also proved useful for the discovery of workarounds. Outside of this domain, the SWORD framework was applied in an academic finance context, where two workaround types were discovered without using any domain knowledge [8]. These types were then used to improve the underlying process.

As an extension to the framework, active learning, in combination with logistic regression, discovered an additional seven workaround types in a separate healthcare case study [9]. With the SWORD framework forming a strong base for this study, active learning primarily decreased the time and effort necessary for interviews to determine which indicated cases indeed point to workarounds.

4. Conclusion

In this work, we present the SWORD Extractor: an R-based tool that makes using the SWORD framework simpler by helping the user select relevant information from an event log and indicating which patterns are applicable to the data. A straightforward analysis can be performed directly within the tool by inspecting the pattern scores and individual indicated cases. It also supports more complex analyses in R, or any other tool by simply exporting the pattern scores. While this version is complete given the current level of related research, updates to the tool are planned for future work. We will discuss two of them here.

First, in the current version of the SWORD Extractor, all nine patterns used in previous research are available. The full SWORD framework consists of 22 patterns. Although some patterns may depend directly on very specific domain knowledge [6, 7], including other SWORD patterns without such dependencies is a natural extension of the current tool. In addition, the SWORD framework itself is designed to be extendable, so such new patterns may also be added to the tool when they are discovered. To simplify any additions, new patterns can be easily added to the framework by design. It only requires the user to write the core: a function that calculates the pattern scores based on the log structure. This new pattern can then be integrated into the tool with a single function call. For details, we refer to the documentation.

Second, the visualization of the SWORD patterns follows previous work and is sufficient to rank traces and indicate cases that are likely workarounds. In some cases, this may require a detailed inspection of a case. More explanations from the tool about why a case would be ranked highly could speed up this process and help end users, especially those less experienced with reading event logs. There has not been any research into what visualizations are meaningful and understandable, but this would be an improvement that could improve accessibility even further.

Acknowledgments

This publication is part of the WorkAround Mining (WAM!) project with project number 18490 which is (partly) financed by the Dutch Research Council (NWO).

References

[1] S. Alter, Theory of Workarounds, Communications of the Association for Information Systems (CAIS) 34 (2014). doi:10.17705/1cais.03455.

[2] T. Ejnefjäll, P. J. Ågerfalk, Conceptualizing Workarounds: Meanings and Manifestations in Information Systems Research, Communications of the Association for Information Systems (CAIS) (2019) 340–363. doi:10.17705/1cais.04520.

[3] I. Beerepoot, I. van de Weerd, Prevent, Redesign, Adopt or Ignore: Improving Healthcare using Knowledge of Workarounds, in: Proceedings of the 26th European Conference on Information Systems (ECIS), 2018.

[4] I. van de Weerd, P. Vollers, I. Beerepoot, M. Fantinato, Workarounds in retail work systems: prevent, redesign, adopt or ignore?, in: Proceedings of the 27th European Conference on Information Systems (ECIS), 2019.

[5] N. Outmazgin, P. Soffer, A process mining-based analysis of business process workarounds, Software & Systems Modeling (SoSyM) 15 (2014) 309–323. doi:10.1007/s10270-014-0420-6.

[6] W. van der Waal, I. van de Weerd, I. Beerepoot, X. Lu, T. Kappen, S. Haitjema, H. A. Reijers, Putting the SWORD to the Test: Finding Workarounds with Process Mining, Business & Information Systems Engineering (BISE) (2024). doi:10.1007/s12599-023-00846-3.

[7] W. van der Waal, I. Beerepoot, I. van de Weerd, H. A. Reijers, The SWORD is Mightier Than the Interview: A Framework for Semi-automatic WORkaround Detection, in: Business Process Management (BPM), 2022, pp. 91–106. doi:10.1007/978-3-031-16103-2_9.

[8] N. Outmazgin, W. van der Waal, I. Beerepoot, I. Hadar, I. van de Weerd, P. Soffer, From Automatic Workaround Detection to Process Improvement: A Case Study, in: BPM 2023 Forum, 2023, pp. 372–390. doi:10.1007/978-3-031-41623-1.

[9] W. van der Waal, I. van de Weerd, S. Haitjema, T. Kappen, H. A. Reijers, Whetting the SWORD: Detecting Workarounds by Using Active Learning and Logistic Regression, in: Hawaiian International Conference on System Sciences (HICSS), 2024, pp. 3687–3696. doi:10125/106828.

[10] F. Mannhardt, Sepsis cases - event log, Version 1. 4TU.ResearchData. dataset (2016). doi:10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460.