The SWORD is Mightier than the Interview: A Framework for Semi-automatic WORkaround Detection

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

Utrecht University, Utrecht, The Netherlands
{w.g.vanderwaal,i.m.beerepoot,g.c.vandeweerd,h.a.reijers}@uu.nl

Abstract. Workarounds can give valuable insights into the work processes that are carried out within organizations. To date, workarounds are usually identified using qualitative methods, such as interviews. We propose the semi-automated WORkaround Detection (SWORD) framework, which takes event logs as input. This extensible framework uses twenty-two patterns to semi-automatically detect workarounds. The value of the SWORD framework is that it can help to identify workarounds more efficiently and more thoroughly than is possible by the use of a more traditional, qualitative approach.
Through the use of real hospital data, we demonstrate the applicability and effectiveness of the SWORD framework in practice. We focused on the use of three patterns, which all turned out to be applicable to the characteristics of the data set. The use of two of these patterns also led to the identification of actual workarounds. Future work is geared to the extension of the patterns within the framework and the enhancement of techniques that can help to identify these in real-world data.

Keywords: Workarounds · Automated Detection · Event data · Healthcare · Process Mining · Business Process Analysis.

1 Introduction

Many organizations use standard operating procedures to streamline their work. When procedures are clear, people know what to do. Still, it often happens that work is performed in a way that is different from the prescribed procedure. When confronted with unexpected situations, limited time, or a lack of resources, workers may be unable to follow a procedure and may feel compelled to perform a workaround to solve a problem [13].

Some workarounds are beneficial and can be leveraged to improve organizational procedures [2,6]. In other cases, not following a procedure may be harmful or outright dangerous. Workarounds can result in noncompliance, privacy issues, or negative effects in the process downstream [15, 16]. Whatever the effect, it is important that process owners are provided with insights into the occurrence of workarounds. These insights can help to prevent workarounds from happening again or to improve the concerned procedure [6]. In addition, being able to

structurally and comprehensively identify workarounds would allow for the monitoring of their emergence, diffusion, and evolution, potentially enabling process analysts to detect and respond to new workarounds faster than with current techniques. This motivates our focus on workaround detection.

To date, most studies in which workarounds were identified and analyzed relied on qualitative methods, primarily through interviewing and observing users during their work [6]. This approach has led to valuable insights related to the mechanics and effects of workarounds, as well as the motivations of the people using them [3,15,32]. However, the use of qualitative methods is labor-intensive; furthermore, users may not disclose their normal behavior when they are aware of being observed [33]. Similar to how process mining is used to solve an otherwise time-consuming problem [1], our focus is on the use of event logs and other quantitative analysis techniques.

In this paper, we introduce the Semi-automated WORkaround Detection (SWORD) framework. The automated part of the SWORD framework uses 22 patterns to identify potential workarounds from an event log. Of these patterns, 16 are based on existing literature, while the remaining six patterns are new. Whether any pattern can be used in a particular situation is dependent on the characteristics of the data in the event log at hand. While the detection of potential workarounds can be performed in a highly automated fashion, the actual confirmation of the occurrence of workarounds still needs to be done by domain experts. This proposed approach can be expected to partly mitigate the change in behavior that people might exhibit when they are aware of being observed because the data is collected in a non-obtrusive way. Also, since the SWORD framework automates the analysis of event data, it is less labor-intensive to use than finding workarounds through interviews.

In earlier work, we already established that event logs from a Health Information System (HIS) can be used to automatically detect and monitor known workarounds [5]. We stay in line with our earlier focus on the healthcare domain with this present work. Studies have shown that in hospital settings specifically workarounds are a widespread phenomenon [28]: Nurses share each other's passwords to save time, physicians send each other X-rays via WhatsApp to get quick second opinions, and secretaries use shadow systems on paper to track the department's occupation. The principles behind the SWORD framework are nonetheless transparent and we expect that many of the patterns can be transferred to other domains.

To show the potential of the SWORD framework, we apply three of the patterns to real hospital data in the setting of an illustrative case study. The data was obtained through our cooperation with the University Hospital Utrecht (UMCU). Our evaluation shows that the proposed approach is feasible in the sense that the patterns allow for an automated analysis of the data. Furthermore, the use of the SWORD framework was helpful to identify actual workarounds in medical practice.

The structure of this paper is as follows. In Section 2, we will first clarify what workarounds are and to what extent they can be detected, according to

related work. We will explain in more detail how we analyzed the literature as well as the set-up of our case study in Section 3. The results of these steps are reported upon in Section 4. Finally, we will discuss the implications of our work in Section 5.

2 Related Work

Workarounds occur when users intend to reach a goal but perceive a block to do so using the official procedure [13]. Similar to the conformance-checking field of process mining, we can look for differences between process variants to detect them [23]. However, we see two important differences. First of all, a workaround requires an *intent* to reach a business goal. So, fraud, deception, and errors are not in scope. Secondly, the user is unable to achieve this goal by using the intended procedure [13]. So, accidentally following a different route is not a workaround.

In addition, while non-conformance usually supposes strict rules [18,22] or a known process model to conform to [29], workarounds can occur without these. Deviance mining also uses process mining but looks at smaller deviations between processes [24], which may also be useful to mine for workarounds. Quite different from a deviance, a workaround can be very common. If a workaround is sufficiently effective, it may be shared throughout the organization, potentially becoming more common than the official procedure [17].

One approach specifically used to automatically detect workarounds is to use deep learning [33]. Neural networks are trained to recognize different workaround types from event logs. These methods can be difficult to use in practice because they require a large amount of labeled training and testing data. In addition, even if neural networks reach a high classification accuracy, it is difficult to explain why this is happening [26], which is often required if you want to use the results in a healthcare environment.

Outside of the control-flow, the time, resource, and data perspectives are valuable for conformance checking [23]. By investigating workarounds discovered using qualitative methods, such as interviews and observations, previous studies show that these perspectives can also be used to recognize different types of workarounds [5]. We will continue this multi-perspective approach to investigate if, in addition to recognizing workarounds, we can discover new workarounds using event logs.

3 Research Method

Our research method consists of two phases: a phase in which we define a list of workaround detection patterns and a phase in which we test them. The two phases and their underlying activities are depicted in Figure 1. We will describe the phases in more detail in this section.

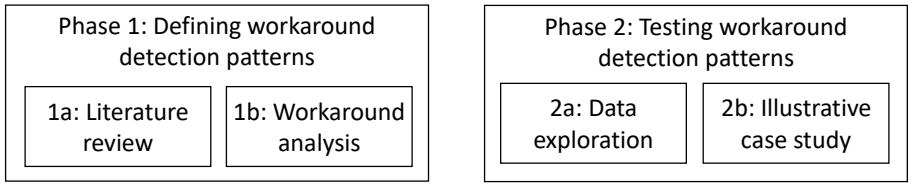


Fig. 1: The phases of our research method

3.1 Phase 1: Defining the workaround detection patterns

As described in the previous section, detecting workarounds has similarities with several approaches in process mining, such as conformance checking. Both focus on differences between the intended and actual process. To investigate these approaches, we carried out a literature review to collect an overview of process mining approaches that can be applied to workaround detection.

After an initial literature search, we selected five (systematic) literature reviews as the starting point of our own study. The reviews focused on several (sometimes overlapping) process mining topics, namely conformance checking [12], process variant analysis [31], predictive process monitoring [11], deviance mining [24], and process mining in healthcare [4]. In the next step, we selected relevant papers from these reviews. We used the following inclusion criteria: (1) the described approach focuses on the differences between process variants, from the control-flow, data, resource, or time perspective, and (2) the described approach uses event data for their analysis.

For example, we did not include supervised learning methods because it is not feasible to label all traces with a workaround or normative label. Note, also, that we keep to the main pattern in situations where slightly differing variants exist. For example, there are multiple version of trace alignment; we do not distinguish between these here. Discovered papers were also searched for new references using reverse snowballing until no new detection patterns came up. Overall, we analyzed 37 papers in detail and included 12 papers in our final analysis, covering 16 detection patterns. The result of this activity is presented in Table 2.

Second, we carried out an analysis of 81 workarounds. These were gathered in previous studies that have been carried out in five different healthcare organizations [5]; a general hospital, two district hospitals, and two specialized care centers. The authors were able to detect multiple discovered workarounds using quantitative methods in a completely different top clinical hospital. This shows that we can expect similar behavior in other healthcare organizations. They are documented as 'workaround snapshots' and include a description of (1) the setting in which the workaround was found, (2) the workaround compared to the normative process, (3) a motivation, and (4) the expected effect of the workaround on cost, time, quality, and flexibility. For an example, see Table 1.

Table 1: Example snapshot

Setting	At a urology clinic. Before a nurse administers medication to a patient, she checks with another nurse to see it is the correct medicine/dosage.
Workaround	The nurses do not register the verification check in the HIS.
Motivation	Registering the check takes a lot of time, so the nurses only do this check vocally.
Effect	This workaround saves time and thus costs, but the data quality is lower. Since the information is not registered, it may lead to errors at a later time.

Using the comparison between the workaround and the normative process, we determined if and how each workaround could be monitored using (event log) data. Similar detection patterns were grouped together. This workaround analysis yielded another six patterns, which have been added to the list in Table 2.

3.2 Phase 2: Testing the workaround detection patterns

In the second phase of our study, we evaluated the list of workaround detection patterns. We followed a two-step approach for this. First, we wanted to establish whether the data necessary to detect a pattern was stored in the HISs. Therefore, we analyzed the data structure of the tables in which the event data of the relevant processes and workarounds are stored. For each pattern, we searched for a table containing the required data to identify it. At this point, we considered if we could use that data to discover new workarounds, or if the pattern relied on specific knowledge and could only be used to monitor known workarounds.

Second, we conducted an illustrative case study in which we took three of the identified workaround detection patterns and tried to find them in real data. We selected patterns that could be applied to the available data and with which we expected to find meaningful differences. The goal of this step was to confirm whether our approach would work on real data. At the same time, the IT department of the University Medical Center (UMC) Utrecht was exploring to implement process mining techniques to better facilitate quality improvement projects and consulted us for our expertise. We then performed a technical proof-of-concept on deploying process mining techniques to detect workarounds using the data from the UMC Utrecht. This academic hospital cares for more than 200,000 patients and has around 12,000 employees. SQL was used to capture the event data that we used in our analyses from the HIS used by the UMC Utrecht (HiX, ChipSoft, Amsterdam). We pseudonymized the data as early as possible by assigning random unique values to all patient, resource, and hospitalization identifiers. After data extraction, we used R for further analysis and visualization.

Time between activities out of bounds The first pattern covers one of the most common workarounds: batching. When taking patient measurements, hospital staff should register the results directly in the HIS, before doing anything else. Instead, they often take measurements from multiple patients in a row, write the results down on a note and register all patient data afterward. This behavior can be detected using the "Time between activities" pattern.

All manually entered patient measurements in the hospital, except for data from the intensive care unit and operating room, are stored in a single table. We extracted the patient ID, resource ID, and registration timestamp of all measurements between two defined moments. To ensure the right events are captured, directly subsequent measurement registrations of the same patient by one resource are removed. Not doing so could result in counting multiple consecutive registrations for the same patient, which would obviously not be batching.

We compared the three shifts of an average, regular Tuesday, which had some overlap. The day shift ran from 7:00h until 16:00h, the evening shift from 15:00h until 0:00h, and the night shift from 23:00h until 8:00h. There were 1403 measurement events, covering 942 unique patients and 460 resources during the day shift, 785 measurement events, 488 unique patients and 320 resources during the evening shift, and 371 measurement events, 245 unique patients and 132 resources during the night shift.

Long duration between time of event and time of logging The second workaround covers the registration delay by comparing the difference between the time of activity and the time of registration. This data is registered in the same measurements table we used during the batching workaround. We again used the patient ID and timestamps of the registration. We also extracted the time of activity and the type of measurement, to see if there are differences in delays between types. Note that the time of activity is generally registered manually, so it is unlikely these are exact values.

We have checked this over the same day we used for the first pattern. This time, we did not separate the three shifts, so we included all measurements that occurred between 7:00h on day one and 8:00h on day two. A single measurement registration could cover multiple measurement types. For example, a registration could cover both a length and weight measurement at the same time. In those cases, we counted these as a separate registration for each measurement type, all using the same timestamps. In total, there were 10118 measurements by 638 resources, covering 1032 unique patients and 204 different measurement types.

Activities executed by a single resource The final workaround we investigated concerns resources that stay the same, while they should have differed over the trace. During the triage process at the emergency department (ED), both a nurse and a physician should see the patient. The main ED table in the HIS logs both the "seen by nurse" and "seen by physician" activities. We

also extracted the resource logging the activities and ED registration identifiers, which is unique for every ED registration and makes for a good case ID.

We have used the data covering one year. This contained 5993 ED registrations. Both the "seen by nurse" and the "seen by physician" activities were logged 5508 times (92%). In 5460 cases (91%) both had seen the patient.

4 Results

4.1 Workaround Detection Patterns – Literature Review

Researchers in process mining fields other than workaround mining, such as conformance checking, deviance mining, predictive process monitoring, and process performance analysis, have developed patterns to detect differences between similar process variants. Since we can recognize different workarounds using varying perspectives, we have structured our review the same way. We start with control-flow and follow this with the data, resource, and time perspective.

The control-flow perspective is used in most fields, often by comparing traces to process models [5,8,27]. Alternatively, some activities should never co-occur [8] or should occur close to specific others [21]. Some events on their own can already be interesting to monitor [7, 9, 24]. Repeated behavior can also be an indication something is going wrong. We can look at how often activities repeat [8,9,24,30] or if there are loops in a trace [9, 20, 24, 30, 34].

We find the most use of the data perspective in the conformance checking field, where we can simply look at the values of data objects [5, 8–10, 19, 31]. If they deviate too much, this can show unintended behavior in the trace. Alternatively, the exact value might not be important, but the value should not change during the trace [8, 10, 19].

The resource perspective shows similar patterns. Some events should always be executed by a specific resource or it needs to stay the same during (part of) the trace [5, 19]. For example, certain medications should only be prescribed by a physician. While not a detection pattern in itself, earlier mentioned patterns can also be used using a resource as case ID. In this way, we can investigate the behavior of resources. For example, if we do so and notice a resource is repeating the same activity often, something might be going wrong [30].

We can find deviating patterns using the time perspective in multiple fields. From a conformance checking viewpoint, some activities may need to be executed at a specific time [19]. Process performance analysis naturally takes time into account too. We can use the time of activity since the start of the trace to predict the performance of the entire trace [7]. Multiple fields distinguish between process variants by looking at the time between activities [10, 19, 30, 34], the duration of a single activity [30, 31, 34], or the total time of a trace [31].

Table 2 shows an overview and description of all 22 detection patterns.

Table 2: Workaround detection patterns

	Detection pattern	Explanation	Reference
Control-flow	Occurrence of an activity	A specific activity occurs	[7, 9, 24]
	Occurrence of recurrent activity sequence	A recurrent activity sequence occurs within a trace	[9, 20, 24, 30, 34]
	Frequent occurrence of activity	An activity frequently occurs within a trace	[8,9,24,30]
	Occurrence of activities in an order different from process model	The order of activities in a trace is other than in a predefined process model	[8, 27]
	Occurrence of mutually exclusive activities	Specific activities occur that are mutually exclusive within a trace	[8]
	Occurrence of unusual neighboring activities	An activity is directly followed by an activity other than usual	[21]
	Occurrence of directly repeating activity	An activity is immediately repeated within a trace	
	Missing occurrence of activity	A specific activity is missing in the trace	
Data	Data object with value outside boundary	The value of a data object deviates from the usual values	[8–10, 19, 31]
	Change in value between events	Data values change unexpectedly between events	[8, 10, 19]
	Specific information in free-text fields	Information is logged in free-text fields instead of dedicated fields	
Resource	Activity executed by unauthorized resource	An activity is executed by a resource other than those authorized	[19]
	Activities executed by multiple resources	Activities within the same trace are executed by multiple resources	[19]
	Activities executed by a single resource	Activities within the same trace are all executed by the same resource	[30]
	Frequent occurrence of activity for a resource	An activity occurs more frequently for one resource compared to other resources	
	Frequent occurrence of value for a resource	A data value occurs more frequently for one resource compared to other resources	

Continued on next page

Table 2: Workaround detection patterns (Continued)

	Detection pattern	Explanation	Reference
Time	Occurrence of activity outside of time period	An activity occurs outside of the usual time period	[19]
	Delay between start of trace and activity is out of bounds	There is a deviation in the delay between the start of the trace and the time of an activity	[7]
	Time between activities out of bounds	There is a deviation in the time between activities	[10, 19, 30, 34]
	Duration of activity out of bounds	There is deviation in the duration of an activity	[30, 31, 34]
	Duration of trace out of bounds	There is a deviation in the duration of a trace	[31]
	Delay between event and logging is out of bounds	There is a deviation in the delay between time of event and time of logging	

4.2 Detection pattern analysis

To investigate what data is required to detect each workaround detection pattern, we have used the data structure of HiX. Using this specific HIS, we explored which columns in the data we would need to use to find each pattern.

To apply these patterns, it should always be clear to which trace an event belongs. Depending on the available data, we can use timestamps (e.g., by grouping events that are temporally close), specific activities (e.g., a trace starts with logging in and ends with logging out), or dedicated case IDs.

Table 3 shows an overview of the required data. We distinguish between four data types that may be required: activity, time, data, and resource. Each detection pattern can require a different level of quality for these types.

- Some patterns need *specific* data. This data must be known beforehand. E.g., a data field may require a certain value. Because of the huge number of data fields, it is not feasible to find these values automatically. We cannot find new workarounds with these patterns, only monitor discovered ones.
- We generally require *high-quality* data. Activity names should be distinguishable from each other, timestamps need to determine when an event happened, data needs to be complete, or we need to know which resource executed the event. The exact requirements differ per process. E.g., for one case, "register measurement" is a good activity name, but for another, we need the measurement type.
- For time, *low-quality* data may be sufficient if high-quality is not available. In that case, timestamps only need to be precise enough to determine a correct event order.

– Some data types are *not be needed* to find a pattern. For example, if we investigate if the right resource performed an activity, we do not require timestamps.

Note that these patterns do not require a specific case ID focus. Patient IDs can be useful to check if people do not repeat work that has already been done by someone else. On the other hand, using resource IDs would show more information about how a single person is working.

Table 3: Workaround detection patterns with the required data. Those marked with * can only be used to monitor known workarounds.

	Detection patterns	**Activity**	**Time**	**Data**	**Resource**
Control-flow	**Occurrence of activity*	*Specific*	-	-	-
	Occurrence of recurrent activity sequence	High Quality	Low Quality	-	-
	Frequent occurrence of activity	High Quality	-	-	-
	**Occurrence of activities in an order different from process model*	*Specific+ Process Model*	*Low Quality*	-	-
	**Occurrence of mutually exclusive activities*	*Specific*	-	-	-
	**Occurrence of unusual neighboring activities*	*Specific*	*Low Quality*	-	-
	Occurrence of directly repeating activity	High Quality	Low Quality	-	-
	**Missing occurrence of activity*	*Specific*	-	-	-
Data	**Data object with value outside boundary*	-	-	*Specific*	-
	Change in value between events	-	Low Quality	High Quality	-
	**Specific information in free-text fields*	-	-	*Specific*	-

Continued on next page

Table 3: Workaround detection patterns with the required data. Those marked with * can only be used to monitor known workarounds. (Continued)

	Detection patterns	Activity	Time	Data	Resource
Resource	Activity executed by unauthorized resource	-	-	-	High Quality
	Activities executed by multiple resources	High Quality	Low Quality	-	High Quality
	Activities executed by a single resource	High Quality	Low Quality	-	High Quality
	**Frequent occurrence of activity for a resource*	*High Quality*	-	-	*Specific*
	**Frequent occurrence of value for a resource*	-	-	*High Quality*	*Specific*
Time	Occurrence of activity outside of time period	High Quality	High Quality	-	-
	Delay between start of trace and activity is out of bounds	High Quality	High Quality	-	-
	Time between activities out of bounds	High Quality	High Quality	-	-
	Duration of activity out of bounds	High Quality	High Quality	-	-
	Duration of trace out of bounds	-	High Quality	-	-
	Delay between event and logging out of bounds	-	High Quality	High Quality	-

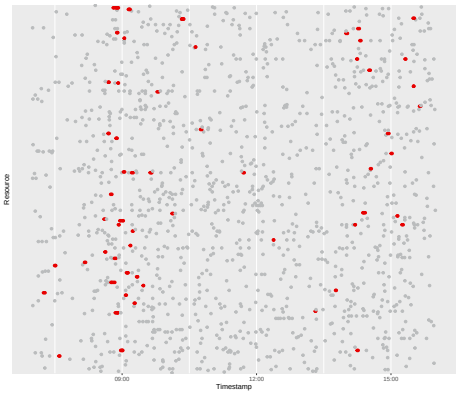
4.3 Illustrative Case Study

We selected three patterns to test if we could apply the SWORD framework: "Time between activities out of bounds", "Delay between event and logging out of bounds", and "Activities executed by a single resource".

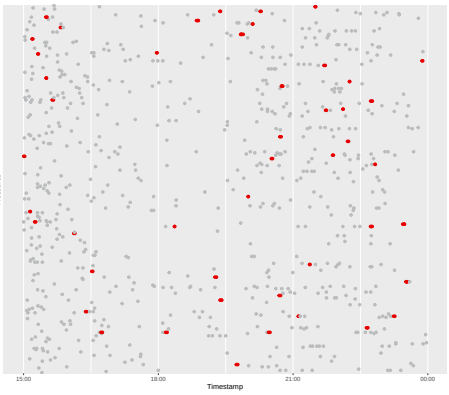
Time between activities out of bounds We tested if we could find the batching workaround in real data. While we are confident that this workaround is likely to be used when measurements in a hospital setting are manually being logged, we do not know where and when it is used in the UMC Utrecht. We can detect this workaround by analyzing the time between events. Since we are interested in resource behavior, we use resources as the case ID. We only need to look at measurement registration events. If the time between the events is very short, there cannot have been enough time to measure a patient, so the employee is most likely practicing batching. This workaround is relatively easy

to recognize and does not require a field expert to do so, allowing us to test the SWORD framework without requiring interviews with them.

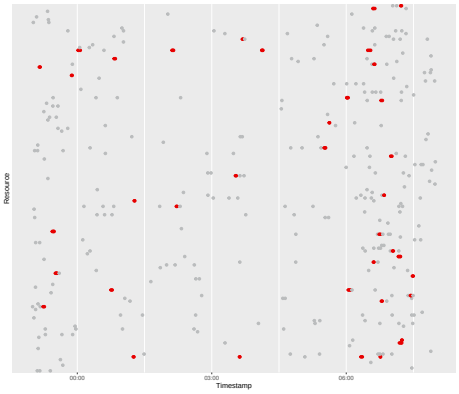
Figure 2 shows graphs containing only manual measurement registration events for the three different shifts in a single day. Every row contains the measurements of a single resource. Since we removed directly subsequent measurements of the same patient, horizontally close events show registrations where there cannot have been enough time to do a new measurement and thus can be considered batching, these are marked in red. We can see that batching occurs more often at the start and end of shifts, which is especially clear during the day shift in Figure 2a.



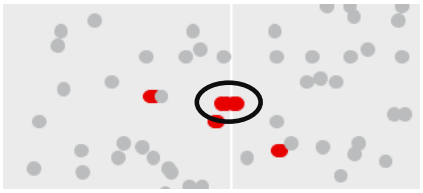
(a) Day shift



(b) Evening shift



(c) Night shift



Time	Patient	Measurements
8:58:07	1	"HR" "NIBP" "Resp" "SpO2" "Temp"
8:59:03	2	"HR" "NIBP" "Resp" "SpO2" "Temp"
8:59:50	3	"HR" "NIBP" "Resp" "SpO2" "Temp"
9:00:24	4	"HR" "NIBP" "Resp" "SpO2" "Temp"
9:00:53	5	"HR" "NIBP" "Resp" "SpO2" "Temp"

(d) Example batching during the day shift. Note that patient IDs are pseudonymized.

Fig. 2: Measurement events per resource for three consecutive shifts. **Grey dots** are normal measurements. **Red dots** are part of batching.

Delay between event and logging out of bounds We compared the time of activity and the time of registration to determine the registration delay. Since there were 204 different measurement types, we limit our results to the most common ten: early warning score (EWS), heart rate (HR), length, non-invasive blood pressure (NIBP), resting pulse (Resp), oxygen saturation (SpO2), temperature (Temp), visual analog scales (VAS), numeric VAS during activity (VASNRSact), and weight. To aid visibility, we filtered out delays of over an hour.

Figure 3 shows the results. Every measurement type has a boxplot showing the delays linked to it. Every type has its own distribution and thus every type has its own time delays that are considered outliers. All outliers can be considered to be different from the common process, but without an expert, we cannot conclude these are workarounds, mistakes, or something else.

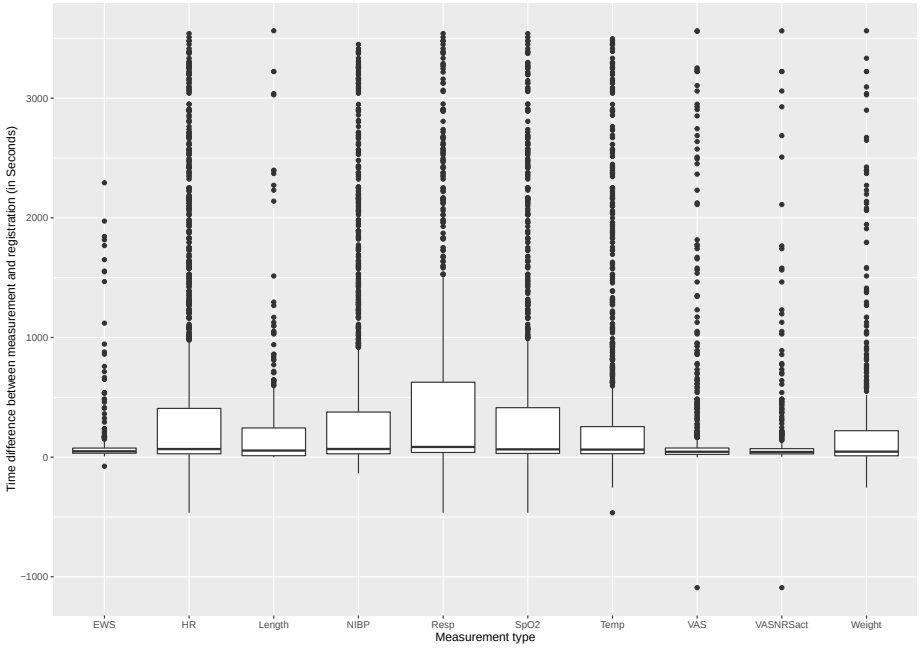


Fig. 3: Boxplot showing the delay from measurement to registration per measurement type

Activities executed by a single resource We compared the resource linked to the "seen by nurse" and "seen by physician" activities within the same ED registration, intending to find cases where both events were logged by one person instead of two. Surprisingly, in *all* 5460 cases with both activities, the resource was the same. This could indicate a structural difference between the prescribed

procedure and the process in practice. The official procedure could also be different than what we expect, so this would require expert input.

Conclusion We have tested three of the detection patterns in our SWORD framework to see if we could use them with real data. We were able to apply all three patterns and found meaningful results in that they either clearly point to workarounds or warrant further investigation.

5 Discussion and Conclusion

In this paper, we have introduced the SWORD framework. With our review, we have discovered twenty-two patterns that are incorporated in this framework. Sixteen of them are based on literature. We determined the remaining six patterns based on previously discovered workarounds.

Our illustrative case study shows that we can detect workarounds using simple data. We successfully used the time between measurements to find batching and we found clear differences in delays between measurements and their registration for different measurement types.

Our framework is based on the mixed methods approach that is used to recognize various known workarounds in data [5]. We use the same perspectives for detection; control-flow, data, resource, and time. The framework points to specific patterns that can be used for each of these perspectives.

Compared to a neural network approach [33], the SWORD framework is more focused. Instead of using a full event log, the patterns have simpler data requirements. They also do not require data to be labeled as a workaround or normative process beforehand. This saves time and effort from experts. Also, we can find workarounds that are not similar to those in the data.

Note that we have only tested three of the twenty-two patterns with real data. While the requirements for remaining patterns are determined using the HIS structure, in practice, the actual data may not fit completely to it. Multiple snapshots from [5] describe this behavior. While some data should be logged in a certain field, users find it easier to log it in free-text fields instead. This can make it difficult to find these patterns. In the future, we will investigate to what extent these workarounds patterns can be detected with real data.

The SWORD framework uses the patterns as singular options to detect differences, but some workarounds can be detected with multiple patterns [5]. To improve detection, we could use machine learning methods, such as classification [25] or clustering [14]. These can combine detection patterns, allowing us to effectively consider processes from multiple angles at the same time.

Acknowledgment

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. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: A survey of issues and approaches. Data & Knowledge Engineering **47**(2), 237–267 (nov 2003). https://doi.org/10.1016/s0169-023x(03)00066-1
2. Alter, S.: Theory of Workarounds. Communications of the Association for Information Systems **34** (2014). https://doi.org/10.17705/1cais.03455
3. Azad, B., King, N.: Enacting computer workaround practices within a medication dispensing system. European Journal of Information Systems **17**(3), 264–278 (jun 2008). https://doi.org/10.1057/ejis.2008.14
4. Batista, E., Solanas, A.: Process Mining in Healthcare: A Systematic Review. In: 2018 9th International Conference on Information, Intelligence, Systems and Applications. IEEE (jul 2018). https://doi.org/10.1109/iisa.2018.8633608
5. Beerepoot, I., Lu, X., van de Weerd, I., Reijers, H.A.: Seeing the Signs of Workarounds: A Mixed-Methods Approach to the Detection of Nurses' Process Deviations. In: Proceedings of the Annual Hawaii International Conference on System Sciences. Hawaii International Conference on System Sciences (2021). https://doi.org/10.24251/hicss.2021.456
6. Beerepoot, I., van de Weerd, I.: Prevent, Redesign, Adopt or Ignore: Improving Healthcare using Knowledge of Workarounds. In: Twenty-Sixth European Conference on Information Systems (2018)
7. Bolt, A., de Leoni, M., van der Aalst, W.M.P.: Process variant comparison: Using event logs to detect differences in behavior and business rules. Information Systems **74**, 53–66 (may 2018). https://doi.org/10.1016/j.is.2017.12.006
8. Borrego, D., Barba, I.: Conformance checking and diagnosis for declarative business process models in data-aware scenarios. Expert Systems with Applications **41**(11), 5340–5352 (sep 2014). https://doi.org/10.1016/j.eswa.2014.03.010
9. Bose, R.P.J.C., van der Aalst, W.M.P.: Discovering signature patterns from event logs. In: 2013 IEEE Symposium on Computational Intelligence and Data Mining. IEEE (apr 2013). https://doi.org/10.1109/cidm.2013.6597225
10. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. Expert Systems with Applications **65**, 194–211 (dec 2016). https://doi.org/10.1016/j.eswa.2016.08.040
11. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Milani, F.: Predictive Process Monitoring Methods: Which One Suits Me Best? International conference on business process management pp. 462–479 (Apr 2018). https://doi.org/10.1007/978-3-319-98648-7_27
12. Dunzer, S., Stierle, M., Matzner, M., Baier, S.: Conformance checking: A state-of-the-art literature review. In: Proceedings of the 11th International Conference on Subject-Oriented Business Process Management. ACM Press (2019). https://doi.org/10.1145/3329007.3329014
13. Ejnefjäll, T., Ågerfalk, P.J.: Conceptualizing Workarounds: Meanings and Manifestations in Information Systems Research. Communications of the Association for Information Systems pp. 340–363 (2019). https://doi.org/10.17705/1cais.04520
14. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Proceedings of the 2nd International Conference on KnowledgeDiscovery and Data Mining (1996)
15. Ignatiadis, I., Nandhakumar, J.: The effect of ERP system workarounds on organizational control: An interpretivist case study. Scandinavian Journal of Information Systems **21**(2), 3 (2009)

16. Ilie, V.: Psychological reactance and user workarounds. a study in the context of electronic medical records implementations. ECIS (2013)
17. Lauer, T., Rajagopalan, B.: Examining the relationship between acceptance and resistance in system implementation. AMCIS 2002 Proceedings p. 179 (2002)
18. Lazovik, A., Aiello, M., Papazoglou, M.: Associating assertions with business processes and monitoring their execution. In: Proceedings of the 2nd international conference on Service oriented computing. ACM Press (2004). https://doi.org/10.1145/1035167.1035182
19. de Leoni, M., van der Aalst, W.M.P.: Aligning Event Logs and Process Models for Multi-perspective Conformance Checking: An Approach Based on Integer Linear Programming. In: Lecture Notes in Computer Science, pp. 113–129. Springer Berlin Heidelberg (2013). https://doi.org/10.1007/978-3-642-40176-3_10
20. Lo, D., Cheng, H., Han, J., Khoo, S.C., Sun, C.: Classification of software behaviors for failure detection. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM Press (2009). https://doi.org/10.1145/1557019.1557083
21. Lu, X., Fahland, D., van den Biggelaar, F.J.H.M., van der Aalst, W.M.P.: Detecting Deviating Behaviors Without Models. In: Business Process Management Workshops, pp. 126–139. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-42887-1_11
22. Mahbub, K., Spanoudakis, G.: A framework for requirents monitoring of service based systems. In: Proceedings of the 2nd international conference on Service oriented computing. pp. 84–93. ACM Press (2004). https://doi.org/10.1145/1035167.1035181
23. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Balanced multi-perspective checking of process conformance. Computing **98**(4), 407–437 (feb 2015). https://doi.org/10.1007/s00607-015-0441-1
24. Nguyen, H., Dumas, M., La Rosa, M., Maggi, F.M., Suriadi, S.: Business Process Deviance Mining: Review and Evaluation. arXiv preprint arXiv:1608.08252 (Aug 2016)
25. Osuna, E., Freund, R., Girosi, F.: An Improved Training Algorithm for Support Vector Machines. In: Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop. pp. 276–285. IEEE (1997). https://doi.org/10.1109/nnsp.1997.622408
26. Ras, G., van Gerven, M., Haselager, P.: Explanation Methods in Deep Learning: Users, Values, Concerns and Challenges. In: The Springer Series on Challenges in Machine Learning, pp. 19–36. Springer International Publishing (2018). https://doi.org/10.1007/978-3-319-98131-4_2
27. Rebuge, Á., Ferreira, D.R.: Business process analysis in healthcare environments: A methodology based on process mining. Information Systems **37**(2), 99–116 (apr 2012). https://doi.org/10.1016/j.is.2011.01.003
28. Röder, N., Wiesche, M., Schermann, M.: A Situational Perspective on Workarounds in IT-Enabled Business Processes: a Multiple Case Study. In: ECIS (2014)
29. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1), 64–95 (mar 2008). https://doi.org/10.1016/j.is.2007.07.001
30. Swennen, M., Martin, N., Janssenswillen, G., Jans, M., Depaire, B., Caris, A., Vanhoof, K.: Capturing Resource Behaviour From Event Logs. In: SIMPDA. pp. 130–134 (2016)

31. Taymouri, F., La Rosa, M., Dumas, M., Maggi, F.M.: Business process variant analysis: Survey and classification. Knowledge-Based Systems **211**, 106557 (jan 2021). https://doi.org/10.1016/j.knosys.2020.106557
32. Tucker, A.L.: The Impact of Workaround Difficulty on Frontline Employees' Response to Operational Failures: A Laboratory Experiment on Medication Administration. Management Science **62**(4), 1124–1144 (apr 2016). https://doi.org/10.1287/mnsc.2015.2170
33. Weinzierl, S., Wolf, V., Pauli, T., Beverungen, D., Matzner, M.: Detecting Workarounds in Business Processes-a Deep Learning method for Analyzing Event Logs. In: Proceedings of the 28th European Conference on Information Systems (May 2020)
34. Wynn, M.T., Poppe, E., Xu, J., ter Hofstede, A.H.M., Brown, R., Pini, A., van der Aalst, W.M.P.: ProcessProfiler3D: A visualisation framework for log-based process performance comparison. Decision Support Systems **100**, 93–108 (aug 2017). https://doi.org/10.1016/j.dss.2017.04.004