Capturing Process Behavior with Log-Based Process Metrics

Peer-reviewed author version

SWENNEN, Marijke; JANSSENSWILLEN, Gert; JANS, Mieke; DEPAIRE, Benoit & VANHOOF, Koen (2015) Capturing Process Behavior with Log-Based Process Metrics. In: Ceravolo, Paolo; Rinderle-Ma, Stefanie (Ed.). Proceedings of the 5th International Symposium on Data-driven Process Discovery and Analysis, p. 141-144.

Handle: http://hdl.handle.net/1942/20239

Capturing Process Behavior with Log-Based Process Metrics

Marijke Swennen[1], Gert Janssenswillen[1,2], Mieke Jans[1], Benoît Depaire[1], Koen Vanhoof[1]

[1]Hasselt University, Agoralaan Gebouw D, 3590 Diepenbeek, Belgium
[2]Research Foundation Flanders (FWO), Egmontstraat 5, 1000 Brussels, Belgium
{marijke.swennen, gert.janssenswillen, mieke.jans, benoit.depaire, koen.vanhoof}@uhasselt.be

Abstract. Currently, process mining literature is primarily focused on the discovery of comprehensible process models that best capture the underlying behavior in event logs. Consequently, the resulting models a) aggregate information, based on algorithm-specific assumptions, and b) transform information into a simplified representation. Both characteristics, which are valuable in certain, different contexts, suffer from the inability to describe objectively the behavior that is inherent to the event log at hand. In this paper, we present an overview of log-based process metrics to capture the process behavior in an event log, without the need to first discover a model. The metrics provide a process owner with unbiased, algorithm-agnostic information of the event log, as a starting point of the process analysis. The constructed metrics also serve as a mean to objectively compare different event logs in terms of time-related and variance aspects.

Keywords: Process mining • Operational excellence • Process behavior • Log-based process metrics

1 Introduction

Process mining is intended to detect strategic insight from business processes by extracting valuable information from event logs. Next to discovering process models from event logs, process mining is also used to check the conformance between a process model and reality and to extend process models with extra information [21]. Starting point for performing a process mining task is an event log. When performing a discovery task on an event log, a process model is extracted without using any additional information [22]. Many discovery algorithms have been introduced [3], [24] and each has its specific assumptions resulting in models not suited for or aimed at describing the behavior that is inherent to the event log objectively and in a detailed fashion.

Looking from a Business Process Management perspective to models, business processes should be modified and improved continuously driven by the continuous improvement concept. This concept is related to methodologies such as lean management, Six Sigma and business process improvement and reengineering [1], [4].

In literature, it has already been suggested that process mining can be used to support operational excellence in companies [21, 22, 23]. However, if companies want to implement methodologies such as lean management or Six Sigma using process mining, it can be cumbersome to decide which discovery algorithms and assumptions to choose. Moreover, process models discovered from an event log are not always perfect representations of the reality. Therefore, we present log-based process metrics, which are measures that indicate how the current process is running, without the need of a process model. In contrast to traditionally used KPIs (for measuring performance), the proposed metrics are constructed on the level of the event log or the activities executed in the event log instead of the output level.

The goal of this paper is to present an overview of useful process metrics that provide an unbiased picture of the present process behavior. Additionally, these metrics can be used to compare different event logs in an objective manner. The different process metrics will be presented in section 2, followed by an illustration in section 3. The implementation and illustrative results will be discussed in section 4. Finally, Section 5 presents the related work and conclusions and future work are presented in section 6.

2 Log-Based Process Metrics

Building on the idea to calculate the distance between two event logs presented in [15], the goal of this research is providing process metrics to identify and quantify the behavior of a process. Based on the findings in literature, four categories of process performance indicators -quality, time, costs and flexibility- have been defined in [8]. In this paper, we will focus on the dimensions *time* and *structuredness. S*tructuredness is chosen, because we want to measure how structured -and not how flexible- the behavior in the event log is. Although structuredness has been defined in [24] as a quality metric to measure the ease of interpretation of a process model, we will define structuredness as the level of variation in the event log.

A list of possible metrics is provided in Table 1 and Table 2. According to the study on model-log evaluation metrics in [2], only one dimension or level of analysis should be measured by each metric, in order to remain comprehensible. Building on the different feature scopes presented in [15], we will assign each metric to one of the following three levels of analysis: (1) *the log level*, which represents the complete event log, (2) *the trace level*, representing characteristics of sequences of activities, and (3) *the activity level*, representing characteristics of the activity types, aggregated over the entire log. Other possible levels, that are out of the scope of this research, are *multiple traces* (characteristics on dependencies between traces) and *multiple activities* (characteristics on dependencies between activities within a trace). All metrics have been defined based on discussions with people from industry.

Table 1. Log-based process metrics of the time dimension

Metric class	Metric	Level of analysis
Duration	Throughput time of the log	Log
	Throughput time of cases: summary statistics of the throughput time of all cases in the log	Log
	Throughput time of a trace: summary statistics of the throughput time of a specific trace	Trace*
Actual processing time	Actual processing time of a trace: summary statistics of the actual processing time of a specific trace	Trace*
	Duration of an activity: summary statistics of the actual processing time of a specific activity type	Activity
Waiting time	Waiting time of a trace: summary statistics of the total waiting time in a specific trace	Trace*

Table 2. Log-based process metrics of the structuredness dimension

Metric class	Metric	Level of analysis
Variance	Number of traces: - absolute number of traces in the log - relative number of distinct traces per 100 cases	Log
	Trace coverage: absolute and relative number of traces that cover 80 % of the log	Log
	Trace frequency: absolute and relative number of times a specific trace occurs in the log	Trace*
	Trace length: summary statistics of the number of activities in each trace	Log
	Trace length: absolute and relative frequency of activities in a specific trace	Trace*
	Activity type frequency: summary statistics of the number of times a specific activity type occurs in a specific trace	Trace*
	Activity type frequency: absolute and relative frequency of activity types in the complete log	Activity
	Activity presence in traces: absolute and relative number of traces where a specific activity type is present	Activity
	Number of distinct start activities: absolute and relative number of activity types that are the first activity of a case	Log
	Number of distinct end activities: absolute and relative number of activity types that are the last activity of a case	Log
	Presence of start activities: absolute and relative number of traces that start with a specific activity type	Activity
	Presence of end activities: absolute and relative number of traces that end with a specific activity type	Activity

Table 2 (cont.). Log-based process metrics of the structuredness dimension

Metric class	Metric	Level of analysis
Self-loops	Number of traces with a self-loop: absolute and relative number of traces that contain one or more self-loop(s)	Log
	Number of self-loops: summary statistics of the number of self-loops within a trace	Log
	Number of self-loops: absolute and relative number of self-loops within a specific trace	Trace*
	Size of self-loops: summary statistics of the size of self-loops in the complete event log (including/excluding activities without self-loops)	Log
	Size of self-loops: summary statistics of the size of self-loops in a specific trace (including/excluding activities without self-loop)	Trace*
	Size of self-loops: summary statistics of the size of self-loops of a specific activity type (including/excluding activities without self-loop)	Activity
Repetitions (excluding self-loops)	Number of repetitions: absolute and relative number of repetitions of an activity in the complete event log	Log
	Number of repetitions: summary statistics of the number of times an activity type is repeated within a specific trace	Trace*
	Number of repetitions: summary statistics of the number of times a specific activity type is repeated within a case	Activity
Batch processing indicator	Frequency of batch processing: absolute and relative number of times batch processing occurs in the complete event log	Log
	Frequency of batch processing: absolute and relative number of times two or more activity instances of the same activity type are executed in a batch	Activity
	Number of activities executed in batch: summary statistics of the number of activity instances of the same activity type executed in a batch	Log

* For reasons of practicality, it might be only of interest to calculate this metric for a predefined number of traces, e.g. only for the top 10 most frequent traces.

3 Illustration of Log-based Process Metrics

To clarify the different metrics presented in Table 1 and Table 2, an example event log will be used. This event log contains 12 cases in which activity instances of 6 different activity types are executed. In total 76 activity instances have been executed (see Fig. 1). The length of the activity instance indicates their duration. For example in case 2, the first activity instance of A takes 2 units of time and the second activity instance of A takes 1 unit of time to be completed.

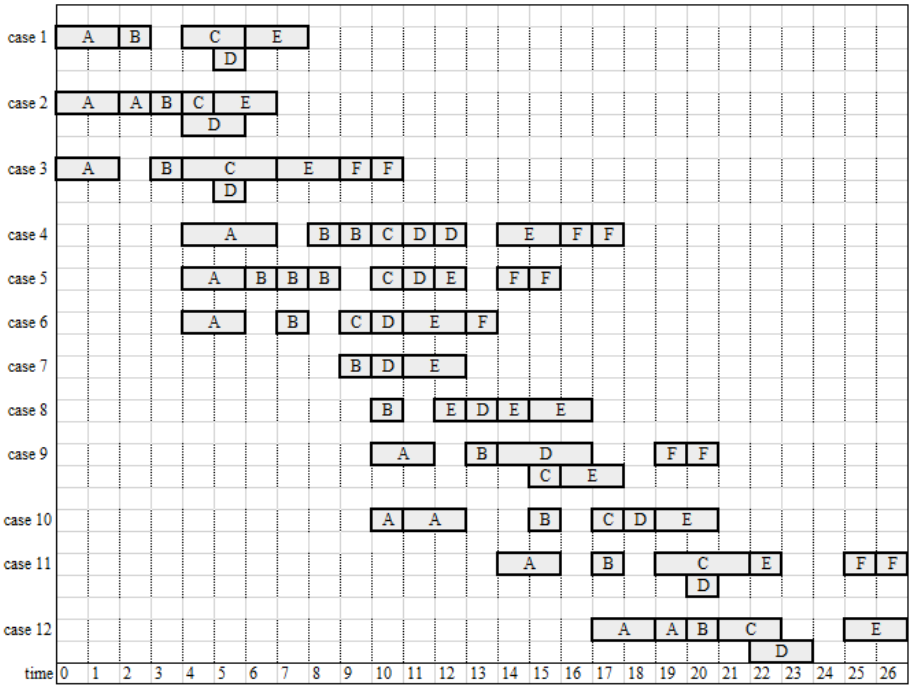

Fig. 1. Example event log

3.1 Time

Duration.

- *Throughput time of the log – log level.* To get a first notion about the complete event log, the sum of the duration of all cases including the idle time in each case can be calculated. This is equal to the total of the throughput time of all cases in the log. The example event log has a throughput time of 118.
- *Throughput time of cases – log level*. The throughput time of a case is the total duration of the case, or the difference between the end timestamp and the start timestamp of the case. Possible idle time is also included in this calculation. For example, the throughput time for case 4 equals 14. Providing this metric on a case level would not increase the user's understanding of the underlying behavior. Therefore, the summary statistics of these throughput times are presented as a metric, to describe the case throughput time in an aggregated fashion. The summary statistics that will be calculated for this and other metrics are the minimum, maximum, median, mean, standard deviation, first quartile, third quartile and interquartile distance. The average throughput time of all cases in the example is 9.83, the standard deviation is 2.86, and the median 10.5. The shortest throughput time is 4 (case 7) and the longest is 14 (case 4).

- *Throughput time of a trace – trace level*. Instead of looking at all cases in the log, it can be interesting to analyze the different process variants or traces in the log. Dividing an event log in homogeneous subsets of traces was presented in [19] in order to overcome the difficulty of analyzing large, unstructured processes. The number of traces in the log will be explained further on as a metric of variance. As a time-related metric we propose the throughput time of a trace which can be calculated for each trace. One example of a trace in the example event log is AABCDE, which is executed in case 2, 10 and 12, with individual throughput times of 7, 11, and 10, respectively. This corresponds to an average throughput time of this trace of 9.33, a standard deviation of 2.08 and a median of 10.

Actual Processing Time.

- *Actual processing time of a trace – trace level*. The actual processing time of a trace is the sum of the actual processing time of all activities that are executed in this trace. In the example, trace ABCDEFF occurs 2 times, in cases 3 and 11. case 3 has an actual processing time of 11 and case 11 has an actual processing time of 10. This results in an average actual processing time for this trace of 10.5.
- *Duration of an activity – activity level*. Finally, if both a start and end timestamp are provided for each activity instance, duration can also be calculated on activity-level. For each activity type, an overview of the actual processing time -or the service time- of this activity type can be of interest. For example, concerning all 13 activity instances of activity type A in our event log, the average duration is 1.85.
  Building from the duration of an activity in the event log, a *bottleneck indicator* would be a useful metric. In a process, a bottleneck is an activity that obstructs other activities to be executed properly and determines the continuation of the whole process [14]. According to the Theory Of Constraints (TOC), introduced by Goldratt in 1984 [5], weak links or constraints should be eliminated from a process because 'a process is only as strong as its weakest link'. Based on this theory is the drum-buffer-rope (DBR) methodology, which is also developed by Goldratt [5]. In a manufacturing context, the "drum", which is the constraint or the bottleneck in the process, determines the pace of the products that move through the system. The "buffer" represents the time between the faster moving products in front and the constraint or slowest product. This buffer is used to protect the constraint and the system from disruptions such as breakdowns. The "rope" finally, makes sure that all products move through the system at the pace of the drum [17, 18]. The bottleneck indicator could be calculated by searching for the activity in the process that has the longest duration compared to the duration of the other activities in the process. The calculation of this metric is however out of the scope of this paper.

Waiting Time.

- *Waiting time of a trace – trace level.* In contrast to the actual processing time, the waiting time, or idle time, in an event log can be an indicator of waste according to the principles of lean management [25]. The total waiting time in the event log is the

sum of all time in each case that is not used. The waiting time for case 1 is 1 unit of time. This metric however aggregates to the trace level. Looking at trace ABCDEFF in the example, we can see that the average waiting time is 2.50 and the standard deviation is 2.12.

Instead of looking at the total waiting time or idle time within a trace, it can be interesting to narrow down to the *waiting time of a specific activity type*, which is the time between the arrival of the activity in the trace and the start of this activity. However, as Leemans et al. [11] suggest, performance measures such as waiting times cannot be measured correctly without the presence of a process model. If we, for example, want to calculate the waiting time of the activity instance of C in case 9, it is not clear from the event log if this activity instance could start after the end of A or after the end of B. Information on the concurrency of activities, and thus a process model, is required to calculate this metric.

3.2 Structuredness

Variance.

- *Number of traces – log level.* A first notion of the structuredness or variance in an event log is the number of patterns, or distinct traces, that are recorded in the event log. In order to have a comprehensive metric, the relative number is stated in 'number of distinct traces per 100 cases'. In the example event log, 9 traces can be observed (see Table 3). In relation to the number of cases in the log, which is 12, this corresponds to 75 distinct traces per 100 cases, indicating a rather low level of structuredness (the higher the ratio, the lower the structuredness).
- *Trace coverage – log level.* The minimum number of traces that is required to cover 80 % of the cases. In Table 3, the traces are sorted based on their relative frequency. To cover 80 % of the 12 cases in the log (9.6 cases, rounded to 10), minimum 7 traces are required: 1 trace with a frequency of 3, 1 trace with a frequency of 2 and 5 traces with a frequency of 1. Only the minimal number of traces, not which traces, is stated, since this is not always straightforward. For instance, in our example, it is not straightforward to choose 5 out of the 7 traces with a frequency of 1.
- *Trace frequency – trace level.* The absolute frequency captures the number of cases that show this trace. The relative frequency expresses the absolute frequency as a proportion of the total number of cases. In Table 3, the frequency of all distinct traces in our example log is provided.

Table 3. Trace frequency of all traces in the example event log

Trace	Absolute trace frequency	Relative trace frequency
A,A,B,C,D,E	3	0.25000000
A,B,C,D,E,F,F	2	0.16666667
A,B,B,B,C,D,E,F,F	1	0.08333333
A,B,B,C,D,D,E,F,F	1	0.08333333
A,B,C,D,E	1	0.08333333
A,B,C,D,E,F	1	0.08333333
A,B,D,C,E,F,F	1	0.08333333
B,D,E	1	0.08333333
B,E,D,E,E	1	0.08333333

- *Trace length – log level.* An overview of the number of activity instances that occur in each trace. In this metric, instances of an activity type, as opposed to activity types, are used. That way, the number of actual transactions on a trace are calculated and aggregated on log-level. For the example event log, on average 6 (rounded from 6.33) activity instances occur per trace with a minimum of 3 and a maximum of 9 activity instances per trace.
- *Trace length – trace level.* This metric shows the number of activity instances executed in each trace. Similar to the previous metric, calculations are done for instances of activity types, as opposed to the activity types themselves. In the example event log, trace ABBCDDEFF contains 9 activity instances of 6 different activity types. To make this number relative, an interesting nominator can be the average trace length of the traces that cover 80 % of the log. Because it is not in every log straightforward which traces exactly cover 80 %, the metric should be calculated on a percentage of the traces that can be identified unambiguously (and deviates from 80 if necessary). In our example, the trace length can either be compared to the average trace length of the top 41.67 % of the event log (traces AABCDE and ABCDEFF), or to the average trace length of the complete event log (since we cannot distinguish unambiguously which trace of frequency 1 is included in the 80 %). This results in a relative number of 1.41 or 1.42 for trace ABBCDDEFF, suggesting that this trace is rather long.
- *Activity type frequency – trace level.* The number of times a specific activity type occurs in a trace. In trace ABBCDDEFF, each activity type appears on average 1.5 times in the trace. The standard deviation is 0.55 and the median is 1.5.
- *Activity type frequency – activity level.* The frequency of a specific activity type in the log. In our example, 15 activity instances of activity type B are executed, which accounts for 19,74 % of the complete log (=15/76).
- *Activity presence in traces – activity level.* For each activity type, the number of traces where the activity type is present. Activity type D in the example event log is present in 9 traces, which is 100 % of all traces in the log, whereas activity type A occurs in 7 out of 9 traces, a presence of 77.78 %.
- *Number of distinct start activities – log level.* The number of activity types that are the first activity instance in one or more of the cases. 2 out of 6 activity types, 33.33 %, are performed as a start activity in the example event log.

- *Number of distinct end activities – log level.* The number of activity types that are the last activity instance in one or more of the cases. 2 out of the 6 activity types, 33.33 %, are the final activity in the example event log.
- *Presence of start activities– activity level.* For each activity type, the absolute and relative number of cases that start with this activity type. The relative number is calculated as a portion of the number of cases, being the number of 'opportunities' that an activity type could be the start activity. For example, activity type B is a start activity in 2 cases, representing a presence of 16.67 % (=2/12). Activity type A is the start activity in the remaining 10 cases (or 83.33 %).
- *Presence of end activities – activity level.* Similar to the previous metric, but for the end activity. Half of the cases in the example event log end with activity type E, the other half ends with activity type F. So for both activity types, the metric will hold the values 6 (absolute frequency) and 50 % (relative frequency).

Self-loops.

The key goal of lean management is waste reduction and avoiding non-value adding activities [25]. Activity instances of the same activity type that are executed more than once immediately after each other are in a self-loop (length-1-loop), what might be an indication of not adding value to the process. If an activity instance of the same activity type is executed 3 times after each other, this is defined as a size 2 self-loop. An activity instance not followed by an activity instance of the same activity type, is a size 0 self-loop (no loop). For now, other patterns, such as length-n-loops or frequent episodes [10], are excluded from this research.

- *Number of traces with a self-loop – log level.* The absolute and relative number of traces in the complete event log that contain at least 1 self-loop. Of the 9 different traces in the example event log, 5 traces or 55.56 % contain 1 or more self-loop(s).
- *Number of self-loops – log level.* Summary statistics of the number of self-loops within a trace, calculated on the analysis level of the complete log. As stated earlier, each combination of two activity instances of the same activity type will be counted as one self-loop of this activity instance. Case 5 contains 2 self-loops: 1 for activity type B and 1 for activity type F. On average, each trace in the example contains 1 self-loop. The standard deviation is 1. Not all traces contain a self-loop, so the minimum number is zero and the maximum number of self-loops within one trace is 3.
- *Number of self-loops – trace level.* On trace level, the number of self-loops that occur within each trace. In trace ABBBCDEFF, 2 self-loops are observed, 1 for activity type B, and 1 for activity type F. To calculate the relative number, each self-loop is counted as 1 occurrence (a self-loop dummy), and the other activity instances are counted as 1. This way, we find 6 occurrences in the trace, which are A, BBB, C, D, E and FF. This gives us 2 out of 6 (or 33 %) self-loops within this trace.
- *Size of self-loops – log level*. The size of a self-loop is based on the number of activity instances of the same activity type that is executed after each other without any other activity type in between. Excluding the other activity instances without a self-loop, the average size of a self-loop in the complete event log is 1.08 with a standard deviation of 0.29. The maximum size of a self-loop is 2. However, we can also include

the other activity instances that do not contain a self-loop in the calculation. Then we see that the average size of a self-loop is 0.21 and the standard deviation is 0.45.

- *Size of self-loops – trace level.* If we consider trace ABBBCDEFF, we see that this trace contains 2 self-loops, 1 of size 2 (BBB) and 1 of size 1 (FF). Excluding the other activity instances without a self-loop, we can state that the average size of a self-loop in this trace is 1.5 and the standard deviation is 0.71. The minimum size is 1 and the maximum size is 2. However, we can also include the other activity instances that do not contain a self-loop in the calculation. Then we see that the activity types A, C, D and E contain self-loops of size zero. The average size of a self-loop is then 0.5 and the standard deviation is 0.84.
- *Size of self-loops – activity level.* Finally, the size of self-loops can also be calculated on the activity-level. Activity type B, for example, occurs 2 times in a self-loop (once in case 4 with size 1 and once in case 5 with size 2). The average size of self-loops of activity type B, excluding occurrences of activity type B without self-loops in other cases, is 1.5. However, when we include all 12 occurrences of activity type B in the event log (self-loops of activity type B are aggregated to 1 occurrence), the average size of a self-loop of activity type B is 0.25 with a standard deviation of 0.62.

Repetitions.

Instead of only looking at activity instances of the same activity type that are executed consecutively, the notion of repetitions (hereby excluding self-loops) can also be interesting in the context of process behavior. A repetition might also be an indication of waste, however it should be possible to report on them separately from self-loops. In trace BEDEE in the example event log, 1 repetition is reported, next to 1 self-loop (both on activity type E).

- *Number of repetitions – log level*. The number of repetitions of an activity type within a trace, aggregated on log-level. Only 1 repetition is recorded in the log, which corresponds to 1.32 % of the total number of activities (=1/76).
- *Number of repetitions – trace level*. The number of repetitions within a trace. For the relative number, the nominator is constructed by counting the number of single activity instances and activity instances in self-loop. For trace BEDEE, this results in a nominator of 4 (B, E, D, EE). The relative number of repetitions in this trace is therefore 0.25 (=1/4).
- *Number of repetitions – activity level*. The number of repetitions within a case per activity type. In our example, one repetition on activity type E is reported on the occurrence of 12 cases. This corresponds to a relative frequency of 8.33 %.

Batch processing.

Another form of waste is batch processing, which can be defined as activities piled and handled simultaneously by the same resource [12], [25]. This results in cases wait-

ing to be handled while other cases are handled immediately. This concept is also related to the structuredness dimension of process behavior. The importance of identifying batch processing in event logs is put forward in [13].

- *Frequency of batch processing – log level.* The number of times batch processing occurs in the complete event log. This is the number of times that two or more activity instances of the same activity type in two distinct cases are started at the same time, have the same duration, and are initiated by the same resource. In our example, no information about resources is present. An example of batch processing is activity type A in case 5 and 6. The activity instance of A in case 4 also starts at this time, but this activity instance has a different duration and is therefore not part of the batch. In total, 11 times batch processing is observed in the log. This accounts for 14.16 % of the complete log (=11/76).
- *Frequency of batch processing – activity level.* The number of times batch processing occurs, defined per activity type. Activity type B is 3 times processed in batch (in case 2 and 3; in case 4 and 5; and in case 5 and 6) which corresponds to 20 % of the complete number of occurrences of activity type B (which is 15).
- *Number of activities executed in batch – log level.* This is the number of activity instances of the same activity type that are processed in batch. In the example event log, one of the occurrences of batch processing consists of 3 activity instances of the same activity type (activity instances of A in case 1, 2 and 3) and all other batch processing occurrences consist of 2 activity instances of the same activity type. This results in an average of 2.1 and a standard deviation of 0.30.

4 Implementation and Illustrative Results

All metrics have been implemented as function in the R-package edeaR [8], which stands for Exploratory and Descriptive Event-based data Analysis in R. R is a programming language which is used extensively for the purpose of statistical analysis and data mining, and furthermore contains extensive functionalities for data visualization. EdeaR enables the handling and analysis of event logs within R, and is fully compatible with the existing XES standard [7]. The package is available through github[1], and comes with several vignettes, which provide a illustrative walk-through.

The metrics have been applied to the event log of a utility provider which describes the process of job requests done by citizens. This event log contains 72 854 cases in which activity instances of 30 different activity types are executed. In total 581 926 activity instances have been executed in 12 090 different variants or traces. This results in a relative frequency of 16.59 distinct traces per 100 cases, hinting at a low structuredness of the event log. The most frequent trace in the event log only appears in 5090 cases or 6.99 % of all cases in the event log. The most frequent activity appears in 98.67 % of all cases, while 16 of the 30 different activity types only appear in less than 10 % of all cases in the event log. More than 99 % of the cases start with the same activity type, but the possible ending activity types are much more diverse, hinting at

[1] https://github.com/gertjanssenswillen/edeaR

the presence of a lot of pending cases that have never been finished. Next to this, the metrics also uncover that an activity type of which the utility provider hopes it will happen as little as possible, appears in one third of all cases. Activities that are often repeated or occur in a self-loop are the activities in which the company collects all information about the customer and the job request, hinting at a lot of waste in the early phase of the process.

5 Related Work

Existing process mining metrics are primarily comparing the reality in an event log with a process model. An overview of existing model-log metrics is presented in [2]. These metrics are divided into four categories which are recall, specificity, precision and generality. Although these categories resemble the well-known conformance checking dimensions fitness, precision, generalization and simplicity [16], they are not completely the same. A similar distinction is also presented in [24], where fitness, precision and generalization are defined as accuracy metrics and simplicity and structuredness are seen as comprehensibility metrics. However, no metrics have been defined for analyzing the event log behavior.

The concept of log-based metrics was introduced in the fuzzy mining algorithm in [6]. Next to this, in [15] a framework is defined to determine which is the best process discovery algorithm. *Features* are defined as numerical characteristics of event logs to capture the distance or diversity between two event logs. Also *measures* are presented, which are used to evaluate the performance and quality of discovery techniques. The categories in which the measures can be divided are again simplicity, fitness, precision, generalization and performance. However, in our approach, we present log-based process metrics that are independent from either process discovery algorithms or process models discovered with these algorithms.

6 Conclusions and Future Work

From literature we can infer that plenty of metrics exist for checking the conformance of process models with reality or for measuring the performance of discovery algorithms. However, choosing the right process discovery technique and its specific assumptions can be cumbersome for companies that have dynamic and rapidly changing processes. Moreover, the resulting process models are not suited for or aimed at describing objectively the behavior that is inherent to the event log. Therefore, log-based process metrics are presented, which give a company an objective indication of the behavior in the event log. The presented metrics are structured along two dimensions, which are time and structuredness, and are calculated on one of three analysis levels: log-, trace-, or activity-level. These metrics provide business people with an objective start to look at their processes. All metrics have been discussed with people from industry, implemented in the edeaR-package in R [9] and applied in a real life case study.

However, some challenges and different perspectives can provide an even better indication of the process behavior observed in an event log. First, the notion of resources

can be of interest to see which worker is executing a task. Next to this, an alternative metric on relevance of a trace, other than frequency, would add incremental value to the current metrics. Moreover, other dimensions of behavior can be taken into account to provide business people with an overall view of their business processes. Finally, indicators or metrics should not be considered to be independent from each other. The results of one metric can be the input of or complement other metrics as stated in [8].

References

1. Bigelow, M.: How To Achieve Operational Excellence. Quality Progress, 35(10), 70-75 (2002)
2. De Weerdt, J., De Backer, M., Vanthienen, J.,Baesens, B.: A Critical Evaluation Study of Model-Log Metrics in Process Discovery. In: zur Muehlen, M., Su, J. (eds.) Business Process Management Workshops, pp. 158-169. Springer, Heidelberg (2011)
3. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A Multi-Dimensional Quality Assessment of State-of-the-art Process Discovery Algorithms Using Real-Life Event Logs. Information Systems, 37(7), 654-676 (2012)
4. Drohomeretski, E., Gouvea da Costa, S.E., Pinheiro de Lima, E., Garbuio, P.A.D.R.: Lean, Six Sigma and Lean Six Sigma: an Analysis Based on Operations Strategy. International Journal of Production Research, 52(3), 804-824 (2014)
5. Goldratt, E.M., Cox, J.: The Goal – A Process of ongoing improvement. North River Press Inc., New York (1984)
6. Günther, C.W., van der Aalst, W.M.P.: Fuzzy Mining: Adaptive Process Simplification Based on Multi-perspective Metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.), International Conference on Business Process Management 2007, LNCS, vol. 4714, pp. 328-343, Springer, Heidelberg (2007)
7. Günther, C.W., Verbeek, H.: Xes-standard definition. Tech. rep., Technische Universiteit Eindhoven (2012)
8. Heckl, D., Moormann, J.: Process Performance Management. In: Handbook on Business Process Management 2, pp. 115-135. Springer, Heidelberg (2010)
9. Janssenswillen, G., Swennen, M., Depaire B., Jans, M.: Enabling Event-data Analysis in R - Demonstration. Submitted at the 5th International Symposium on Data-driven Process Discovery and Analysis (SIMDPA), Vienna (2015)
10. Leemans, M., van der Aalst, W.M.P.: Discovery of Frequent Episodes in Event Logs. In: Proceedings of the 4th International Symposium on Data-driven Process Discovery and Analysis (SIMDPA 2014). Milan (2014)
11. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P: Using Life Cycle Information in Process Discovery. In: (in press) Business Process Management Workshops (2015)
12. Martin, N., Depaire, B., Caris, A.: The Use of Process Mining in a Business Process Simulation Context: Overview and Challenges. In: IEEE Symposium on Computational Intelligence and Data Mining 2014, pp 381-388, IEEE (2014)
13. Martin, N, Swennen, M., Depaire, B., Jans, M., Caris, A., Vanhoof, K.: Batch Processing: Definition and Event Log Identification. Submitted at the 5th International Symposium on Data-driven Process Discovery and Analysis (SIMDPA), Vienna (2015)
14. Melton, T.: The Benefits of Lean Manufacturing: What Lean Thinking has to Offer the Process Industries. Chemical Engineering Research and Design, 83(6), 662-673 (2005)

15. Ribeiro, J., Carmona, J., Mısır, M., Sebag, M.: A Recommender System for Process Discovery. In: Sadiq, S., Soffer, P. (eds.) International Conference on Business Process Management 2014, LNCS, vol. 8659, pp. 67-83, Springer, Heidelberg (2005)
16. Rozinat, A., de Medeiros, A.K.A., Günther, C.W., Weijters, A.J.M.M.,van der Aalst, W.M.P: The Need for a Process Mining Evaluation Framework in Research and Practice. In: ter Hofstede, A., Benatallah, B., Paik, H.Y. (eds.) Business Process Management Workshops 2007, LNCS, vol. 4928, pp. 84-89, Springer, Heidelberg (2008).
17. Sale, M. L., Sale, R. S.: Theory of Constraints as Related to Improved Business Unit Performance. Journal of Accounting and Finance, 13(1), 108-114 (2013)
18. Schragenheim, E., Ronen, B.: Drum-buffer-rope shop floor control. Production and Inventory Management Journal, 31(3), 18-22 (1990)
19. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace Clustering in Process Mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) Business Process Management Workshops, LNBIP, vol. 17, pp. 109-120, Springer, Heidelberg (2009)
20. Van Assen, M.F.: Position Paper - Operational Excellence for Services (2011).
21. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011)
22. van der Aalst, W.M.P. et al.: Process Mining Manifesto. In: Business Process Management Workshops, LNBIP, vol. 99, pp. 169-194, Springer, Heidelberg (2011)
23. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.: Replaying History on Process Models for Conformance Checking and Performance Analysis. WIREs Data Mining and Knowledge Discovery, 2(2), 182-192 (2012)
24. vanden Broucke, S.: Advances in Process Mining: Artificial Negative Events and Other Techniques. Ph.D. thesis, KU Leuven (2014)
25. Womack, J., Jones, D.T.: Lean Thinking: Banish Waste and Create Wealth in Your Corporation. Simon and Schuster, London (1996)