Exploiting Localization of Replay Results: Individual and Group Assessment in a Medical Training Process

Alessandro Berti[1 [0000−0003−1830−4013]] and Wil van der Aalst[1 [0000−0002−0955−6940]]

Process and Data Science group, Lehrstuhl für Informatik 9 52074 Aachen, RWTH Aachen University, Germany

Abstract. In the process mining discipline, the aim of conformance checking is to compare a process model with the execution data, signaling process executions that are not following the model. Several techniques exist to perform such a comparison (alignments, token-based replay), mainly focused on the control-flow perspective (i.e., the order of execution of the activities), assessing which activities in the process are executed according to the model, and which activities are executed in an unfit way. In this report, an assessment of deviations on the Conformance Checking Challenge 2019 dataset, that is about a medical training process and includes the log and the process model, is done using not only the control-flow perspective but also information about resources (to assess the individual and the group performance) and the execution time of each activity (to evaluate the performance against other executions of the same activity). To produce this assessment, the process mining library PM4Py has been used. The report concurs for the academic category of the Conformance Checking Challenge 2019, and considers both the student's and the instructor's perspectives.

Keywords: Localized Conformance Checking · Group Assessment · Noise Filtering.

1 Introduction

Process mining [2] is a branch of data science that aims to extract useful information about business processes starting from events stored in an information system. Process mining is a vast discipline and therefore several aspects exist in which processes could be analyzed, including:

- Process discovery, that aims to extract the process schema from the actual execution data. The process model could be expressed in several formalisms (Petri nets, BPMN, Casual nets, transition systems) decorated by performance metrics, by roles and guards.
- Conformance checking aims to compare an event log and a process model in order to discover deviations and obtain diagnostics information [14].

– Process prediction, that aims to predict the future behavior of a process trace having only some events of the trace.

The importance of conformance checking is growing as is illustrated by the new book on conformance checking [5] and the Gartner report which states "we see a significant trend toward more focus on conformance and enhancement process mining types" [7]. Deviations are related to process executions not following the process model (for example, the execution of some activities may be missing, or the activities are not happening in the correct order), and are usually associated to higher throughput times and lower quality levels. Hence it is important to detect them, understand their causes and re-engineer the process in order to avoid such deviations. A preliminary operation in conformance checking is the replay technique, that aims to align the behavior observed in the log with the behavior observed in the model. Different replay techniques have been proposed, like *token-based replay* [15] and *alignments* [5, 4].

The 1st Conformance Checking Challenge 2019 (CCC19) is a co-located event of the 1st International Conference on Process Mining 2019. The challenge provides participants with artifacts stemmed from a real process, and invites them to analyze the conformance between the observed (event log) and expected behaviour (model) of the process, with an emphasis on providing process owners with interpretable and understandable conformance results, in terms of a report. This report, that concurs for the Academic category of the Conformance Checking Challenge 2019, considers both the student's and the instructor's perspectives (see Section 1.3), and is aiming to provide some insights on anomalous executions with regards to the control-flow, the time and the resource perspectives, replying to the following questions:

1. Which are the activities that are more prone to errors? Which resources are more prone to errors, i.e. to execute in a wrong way the process? Are there groups (clusters) of resources that share the same errors?
2. Is it possible to learn, for each activity, a time-performance distribution that is able to tell if a given execution is fit or unfit according to the execution times of the same activity in other process executions?

The approach is supported by the Python process mining library PM4Py[1].

The paper is organized as follows: in the remainder of the Introduction the process, the dataset and the perspectives will be presented. In Section 2, an overview of Petri nets and random variables estimation will be provided. In Section 3, the library PM4Py will be introduced. In Section 4.1, the pre-processing steps operated on the log and on the model will be explained. In Section 4.2, the steps needed to apply the token-based replay and perform throughput analysis will be explained. In Section 4.3, the localization of the replay on the resource attribute will be described (replying to the Question 1 of the report). In Section 4.4, the steps needed to learn for each activity a probability distribution that describes its execution times are provided, along with considerations on noise

[1] The official site of the library is http://www.pm4py.org

and outliers (replying to the Question 2 of the report). In Section 4.5, the replay is performed again (replying to the Question 1 of the report) in order to evaluate the executions without the outliers. Finally, Section 5 concludes the report.

1.1 The Process

The Conformance Checking Challenge 2019 is focused on a medical training process. In particular, the process captures the procedure used for installing Central Venous Catheters (CVC) with ultrasound. Moreover, this process is the one used by doctors for training medical students in this specific task. The CVC procedure refers to installing a catheter (tube) in a central vein, which is then used for delivering liquids, fluids, or medicines to patients. The general idea of the procedure is the following:

- The patient is prepared, the target vein is identified with ultrasound, and the patient is anesthetized.
- A large hollow needle (Trocar), attached to a syringe, is inserted into the vein with the help of ultrasound.
- The Syringe is used for checking for a blood return, implying that the trocar has been correctly installed. If so, the syringe can be safely removed from the trocar.
- A Guidewire is inserted at about 30 cm through the trocar. Then, the trocar is removed.
- The pathway is widen and the wire is used for guiding the insertion of the Catheter.
- Finally, the guidewire is removed and the the final position of the catheter is checked.

The data was collected during a course delivery from an institution that teaches future doctors how to perform a CVC installation. The training is structured as follows:

- Students are taught how to perform the procedure by their instructors
- Students take a first preliminary test on the procedure (PRE)
- Students can practice the procedure at their discretion
- Students take a final post assessment (POST) to show they have acquired the skills required by the procedure.

1.2 Dataset

The dataset provided with this challenge [11] include the event log corresponding to execution of PRE and POST tests performed by 10 students. The dataset has been collected as part of the multidisciplinary research project Process-Oriented Medical Education (POME) conducted by the School of Medicine and the School of Engineering of the Pontificia Universidad Catolica de Chile.

The dataset includes:

- Description: A short slidedeck with the description of the Challenge and pointers to more information.
- Model: A Delphi method was designed to establish clinical consensus for medical procedures (and processes in general), and it was used to define a BPMN model for the Central Venous Catheter installation with ultrasound. The Delphi panel was answered by 13 experts from 3 medical specialities and 8 medical institutions. The dataset contains the model in BPMN notation, Petri Net notation, in PNG, and Natural Text. Models in other formats derived from those are also welcomed in the Challenge.
- Log: A video was recorded of each execution (PRE and POST for each 10 students). A special software was designed to tag the videos with the activity performed, and their initial and final times. Since it is a manual tagging, the data may contain noise. The dataset contains the log in CVS notation, XES notation, and XLSX The log contains 20 cases, 697 events, 29 activities. The log contains the following information per event:
  - CASEID: Trace ID
  - RESOURCE: Student id executing the procedure
  - ROUND: *Pre* if it was performed by prior the practice sessions (PRE), or *Post* if it was after the practice sessions (POST).
  - EVENTID: Event id
  - ACTIVITY: Name of the activity of the process performed.
  - STAGE: Stage of the procedure the activity belongs to (Operator and Patient Preparation, Ultrasound Preparation, Locate Structures, Venous puncture, Install Guidewire, or Install Catheter).
  - START: Time when the activity started
  - END: Time when the activity was finished
  - VIDEOSTART: Time in the video recording when the activity started
  - VIDEOEND: Time in the video recording when the activity finished.

1.3 Perspectives

Being the process in a training course context, we include two perspectives in the conformance report, one for each stakeholder: Students and Instructor.

- Student's Perspective: Primarily interested on their own performance, a student would aim at identifying his/her own mistakes, looking at improving his/her overall execution before the POST test.
- Instructor's Perspective: Aiming at an aggregated view of the performance of the group, an instructor would try to identify common mistakes, similarities and differences exhibit by the group of students, the need for additional training, so on and so forth.

2 Background and Techniques

2.1 Petri Nets

Petri nets are the most widely used process model in process mining frameworks: popular discovery algorithms like the Alpha Miner and the Inductive Miner

(through conversion of the resulting process tree) can produce Petri nets. An accepting Petri net is a Petri net along with a final marking.

Definition 1 (Accepting Petri nets). *A (labeled, marked) accepting Petri net is a net of the form $PN = (P, T, F, W, M_0, M_F, l)$, which extends the elementary net so that:*

- *(P, T, F) is a net (P and T are disjoint finite sets of places and transitions; $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs).*
- *$W : F \to \mathbb{N}$ is an arc multiset, so that the count (or weight) for each arc is a measure of the arc multiplicity.*
- *$M_0 : P \to \mathbb{N}$ is the initial marking[2].*
- *$M_F : P \to \mathbb{N}$ is the final marking.*
- *$l : T \to \sum \cup \{\tau\}$ is a labeling function that assigns to each transition $t \in T$ either a symbol from $\sum$ or the empty string τ.*

The *preset* of a place, $\bullet p$, is the set of all transitions $t \in T$ such that $(t, p) \in F$. The *postset* of a place, $p\bullet$, is the set of all transitions $t \in T$ such that $(p, t) \in F$. The preset and postset of a transition could be defined in a similar way. A transition t is said to be *visible* if $l(t) \in \sum$; is said to be *hidden* if $l(t) = \tau$. If for all $t \in T$ such that $l(t) \neq \tau$, $|\{t' \in T | l(t') = l(t)\}| = 1$, then the Petri net contains *unique visible* transitions; otherwise, it contains *duplicate* transitions. The initial marking is corresponding the initial state of a process execution. Process discovery algorithms may associate also a final marking to the Petri net, that is the state in which the process execution should end. The execution semantics of a Petri net is the following:

- A transition $t \in T$ is *enabled* (it may *fire*) in M if there are enough tokens in its input places for the consumptions to be possible, i.e. iff $\forall s \in \bullet t : M(s) \geq W(s, t)$.
- Firing a transition $t \in T$ in marking M consumes $W(s, t)$ tokens from each of its input places s, and produces $W(t, s)$ tokens in each of its output places s.

2.2 Overview on Token-based Replay

The application of token-based replay is done on a trace of an event log and an accepting Petri net (a Petri net along with an initial and a final marking). The output of the replay operation is a list of transitions enabled during the replay, along with some numbers: c is the number of consumed tokens (during the replay), p is the number of produced tokens, m is the number of missing tokens, r is the number of remaining tokens. At the start of the replay, it is assumed that the tokens in the initial marking are inserted by the environment, increasing p accordingly (for example, if the initial marking consists of one token in one place, then the replay starts with $p = 1$). The replay operation considers, in order, the activities of the trace. In each step, the set of enabled transitions in the current

[2] A marking $M : P \to \mathbb{N}$ is a place multiset.

marking is retrieved. If there is a transition corresponding to the current activity, then it is fired, a number of tokens equal to the sum of the weight of input arcs is added to c, and a number of tokens equal to the sum of the weight of output arcs is added to p. If there is not a transition corresponding to the current activity enabled in the current marking, then a transition in the model corresponding to the activity is searched (if there are duplicate corresponding transitions, then [15] provides an algorithm to choose between them). Since the transition could not fire in the current marking, the marking is modified by inserting the token(s) needed to enable it, and m is increased accordingly. At the end of the replay, if the final marking is reached, it is assumed that the environment consumes the tokens from the final marking, and c is increased accordingly. If the marking reached after the replay of the trace is different from the final marking, then missing tokens are inserted and remaining tokens r are set accordingly.

The following relations hold between p, c, m and r: (1) $p + m = c + r$ (at the end of the replay), (2) $c \leq p + m$, and (3) $m \leq c$. A fitness value could be calculated for the trace as:

$$f_\sigma = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

A fitness value at log level could also be introduced. For each case L_i of the event log L, let c_i be the number of consumed tokens, p_i the number of produced tokens, m_i the number of missing tokens and r_i the number of remaining tokens. Then, the following formula calculates the fitness at the log level

$$f_L = \frac{1}{2}\left(1 - \frac{\sum_{L_i \in L} m_i}{\sum_{L_i \in L} c_i}\right) + \frac{1}{2}\left(1 - \frac{\sum_{L_i \in L} r_i}{\sum_{L_i \in L} p_i}\right)$$

This quantity is different from the average of fitness values at trace level. When, during the replay, a transition corresponding to the activity could not be enabled, and invisible transitions are present in the model, a technique is deployed to traverse the state space and possibly reach a marking in which the given transition is enabled. A heuristic that uses the shortest sequence of invisible that enables a visible task is proposed. This heuristic tries to minimize the possibility that the execution of an invisible transition interferes with the future firing of another activity.

A well-known problem for token-based replay is the *token flooding problem* [5]. Indeed, when the case differs much from the model, and a lot of missing tokens are inserted during the replay, it happens that also a lot of tokens remain unused and many transitions are enabled. This leads to misleading diagnostics because unwanted parts of the model may be activated, and so the fitness value for highly problematic executions may be too high. To illustrate the token-flooding problem consider a process model without concurrency (only loops, sequences, and choices) represented as a Petri net. At any stage, there should be at most one token. However, each time there is a deviation a token may be added resulting in a state which was never reachable from the initial state.

The original token-based replay approach [15] was only implemented in earlier versions of the ProM framework (ProM4 and ProM5). The implementation described in [15] proposes localized metrics on places of the Petri net that help to understand which parts of the model are more problematic. However, the information provided is not enough to perform throughput analysis or root cause analysis [13, 12, 6]. Root cause analysis could be performed in a conformance checking setting by applying decision trees to information related to fit and unfit executions. To improve performance in the original implementation [15], a preprocessing step could be used to group cases having the same trace. In this way, the replay of a unique trace is done once by the token-based replay. Alternatively, more ad-hoc token-based replay approaches were used by the heuristic miner and the genetic miner. In the latter approach the qualities of candidate models are derived. These techniques tend to put multiple dimensions (replay fitness, precision, etc.) into a single fitness measure. The focus is less on providing diagnostics.

2.3 Probability Distributions

A continuous *random variable* $X : \Omega \to E$ is a measurable function from a set of possible outcomes Ω to a set $E \subseteq \mathbb{R}$. The probability that X takes on a value in the measurable set $S \subseteq E$ is written as:

$$P(X \in S) = P(\{\omega \in \Omega | X(\omega) \in S\})$$

where P is the probability measure equipped with Ω. To a continuous random variable X, we could associate a cumulative distribution function $F(x) = P(X \leq x)\ \forall x \in \mathbb{R}$ and a probability density function f such that

$$P(a \leq X \leq b) = \int_a^b f(x)dx$$

In particular, the probability for X to take any single value a (that is $a \leq X \leq a$) is zero, because an integral with coinciding upper and lower limits is always equal to zero, while $f(a)$ may be different from zero. f is also called the *likelihood* function associated to a probability distribution, since it measures how much a value $a \in E$ is likely according to the distribution. The log-likelihood of f given a, $L(f|a)$, is $L(f|a) = \log(f(a))$. Given a finite set of values $V \subset E$, $V = \{a_1, \ldots, a_n\}$, the log-likelihood of f given V, $L(f|V)$, is

$$L(f|V) = \sum_{v \in V} \log(f(v))$$

Some popular probability distributions are the normal distribution (where $E = \mathbb{R}$; characterized by the mean μ and the standard deviation σ^2), the exponential distribution (where $E = \mathbb{R}^+$; it is characterized by the parameter λ. An exponential distribution of parameter λ has mean equal to $\frac{1}{\lambda}$) and the uniform distribution (where $E = [a, b]$; each value in the interval $[a, b]$ has equal density).

Given a finite set of non-negative values $V \subset \mathbb{R}^+ \cup \{0\}$, the parameters of a normal, an exponential and an uniform probability distribution could be estimated in such a way to maximize the log-likelihood of V. For example, the parameters that maximize the log-likelihood of V for a normal distribution are:

$$\hat{\mu} = \frac{1}{|V|} \sum_{v \in V} v \qquad \hat{\sigma} = \frac{1}{|V|} \sum_{v \in V} (v - \hat{\mu})^2$$

For other distributions, similar formula exist for the parameters. Among the normal, exponential and uniform distributions, with the estimated parameters, the one that maximizes the log-likelihood given V is chosen.

```
from pm4py.objects.log.importer.xes import factory as xes_importer
from pm4py.algo.discovery.inductive import factory as inductive_miner
from pm4py.algo.conformance.tokenreplay import factory
  as token_based_replay
from pm4py.evaluation.replay_fitness import factory
  as replay_fitness_factory

# loads the log
log = xes_importer.apply("C:\\running-example.xes")
# discovers an accepting Petri net
net, im, fm = inductive_miner.apply(log)
# applies token-based replay and prints the result
# (for each trace, the output of token-based replay)
aligned_traces = token_based_replay.apply(log, net, im, fm)
print(aligned_traces)
# applies evaluation factory to get replay fitness
fitness = replay_fitness_factory.apply(log, net, im, fm)
print(fitness)
```

Fig. 1: Example PM4Py code to apply token-based replay to a log and an accepting Petri net.

3 PM4Py: a Python Library Implementing Conformance Checking

PM4Py is a process mining library, written in Python. PM4Py is available for installation in Python 3.6 through the command *pip install pm4py* in Windows, Mac OS X and Linux environments. Additional prerequisites, available at the page *http://pm4py.pads.rwth-aachen.de/installation/*, have to be installed (e.g. for Windows, GraphViz and Microsoft Visual Studio 14 C++ compiler are required). In addition to that, the official Github repository *https://github.com/pm4py/pm4py-source* could be cloned to get access to stable and development branches. PM4Py documentation (more than 180 A4 pages) is available at the page *http://pm4py.pads.rwth-aachen.de/documentation/*.

A set of process discovery (Alpha Miner, Inductive Miner), conformance checking (token-based replay, alignments) and enhancement algorithms is provided in the library. The library has been used to support the process mining part

of the "Introduction to Data Science" course of the Process and Data Science department in the RWTH Aachen University (Germany). An example script, that loads a log, calculates a model using Inductive Miner Directly-Follows [8], and does conformance checking, is provided in Fig. 1. A specific branch has been deployed to host the preprocessing changes and the code deployed to perform the conformance checking on the CCC19 dataset[3].

Token-based replay is the focus of this report. The implementation of the token-based replay proposed in [15] has been made more efficient thanks to some improvements, inherited from the alignments implementation in ProM6 [3]:

1. *Post-fix caching*: a post-fix is the final part of a case. During the replay of a case, the couple marking+post-fix is saved in a dictionary along with the list of transitions enabled from that point to reach the final marking of the model. For the next replayed cases, if one of them reaches exactly a marking + post-fix setting saved in the dictionary, the final part of the replay could be retrieved from the dictionary.
2. *Activity caching*: activity caching means saving in a dictionary, during the replay of a case, the list of hidden transitions enabled from a given marking to reach a marking where a particular transition is enabled. For the next replayed cases, if one of them reaches a marking + target transition setting saved in the dictionary, then the corresponding hidden transitions are fired accordingly to enable the target transition.

To address the token flooding problem, which is one of the most severe problems when using token-based replay, we propose several strategies. The final goal of these strategies is to avoid the activation of transitions that shall not be enabled, keeping the fitness value low for problematic parts of the model. The common pattern behind these strategies is to determine *superfluous tokens*, that are tokens that cannot be used anymore. During the replay, f (initially set to 0) is an additional variable that stores the number of "frozen" tokens. When a token is detected as superfluous, it is "frozen": that means, it is removed from the marking and f is increased. Frozen tokens, like remaining tokens, are tokens that are produced in the replay but never consumed. Hence, at the end of the replay $p + m = c + r + f$. To each token in the marking, an *age* (number of iterations of the replay for which the token has been in the marking without being consumed) could be assigned. The tokens with the highest age are the best candidates for removal. The techniques to detect superfluous tokens are deployed when a transition required the insertion of missing tokens to fire, since the marking would then possibly contain more tokens. One of the following strategies can be used:

1. Using a decomposition of the Petri net in semi-positive invariants [9] or S-components [1] to restrict the set of allowed markings. Considering S-components, each S-component should hold at most 1 token, so it is safe to remove the oldest tokens if they belong to a common S-component.

[3] The branch could be retrieved at the URL *https://github.com/Javert899/pm4py-source/tree/ccc19*

2. Using place bounds [10]: if a place is bounded to N tokens and during the replay operation the marking contains $M > N$ tokens for the place, the "oldest" tokens according to the age are removed.

Localizing fitness issues in the process model is an essential step in the provision of more detailed diagnostics (like throughput time or root cause analysis). Some localized information on places have been proposed by the approach described in [15]:

- *Place underfedness*: when missing tokens are inserted in the place during the replay operation of a case, the place is signed as underfed (it has less tokens than needed at some stage) for the specific case.
- *Place overfedness*: when remaining tokens are in the place after the end of the replay of a case, the place is signed as overfed (it has more tokens than needed) for the specific case.

To introduce some localized information at the transition level, it is important to notice that, when the transition is fired during the replay of a case, is possible to register the *current case status*, for example recording all the values of the attributes of the current and of the previous events of the case. The easiest option is to keep a single value for each attribute, that is corresponding to the value of the last occurrence of the given attribute. So, the following localized information could be introduced at the transition level:

- *Transition underfedness*: some tokens needed to fire the transition are missing. It is possible to flag a transition as underfed for the specific case, saving also the status of the case when the transition has been fired.
- *Transition fitness*: the transition could be fired regularly. In this case, it is possible to save the status of the case when the transition has been fired.

It is important also the save information for events with an activity that is not corresponding to any transition in the model. This could be done saving the current case status when such activities happen.

The localized information is useful to compare, for each problematic entity, the set of cases of the log that are fit according to the given entity and the set of cases of the log that are not fit according to the given entity (called "unfit"). In particular, the following questions can be answered:

1. If a given transition is executed in an unfit way, what is the effect on the throughput time?
2. If a given activity that is not contained in the process model is executed, what is the effect on the throughput time?

These questions can be answered by throughput time analysis. Essentially, an aggregation (for example, the median) of the throughput times of fit and unfit cases is taken into account, and the results compared. Usually, transitions executed in an unfit way are corresponding to higher throughput times. The comparison between the throughput time in non-fitting cases and fitting cases permits to understand, for each kind of deviation, whether it is important or not important for the throughput time.

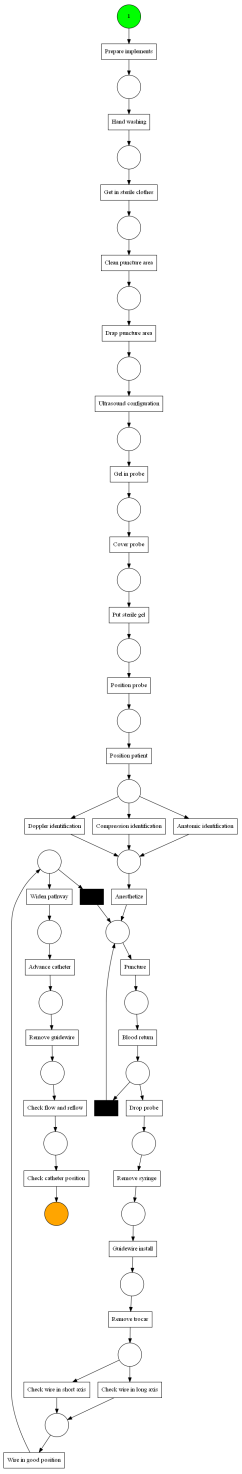

Fig. 2: Representation of the Petri net of the medical process of CCC19.

Table 1: Effects of deviations on the median throughput time on the Pre round log. Median throughput of fit and unfit cases is compared.

Activity	Fit cases	Unfit cases	Fit median throughput time	Unfit median throughput time	Relative throughput
Blood return	8	2	1320	3240	2.45
Anesthetize	2	6	1290	3240	2.51
Put sterile gel	6	4	1260	3240	2.57
Widen pathway	1	9	5640	1380	*0.24*
Puncture	3	7	1380	1560	1.13
Cover probe	6	4	3060	1260	*0.41*
Check wire in short ax	2	6	3210	3060	*0.95*
Gel in probe	3	6	1200	3060	2.55
Drap puncture area	7	3	1380	1560	1.13
Remove trocar	4	6	1290	3060	2.37
Position probe	3	5	1260	4920	3.90
Clean puncture area	4	5	1470	4560	3.10
Get in sterile clothes	1	9	5460	1380	*0.25*
Guidewire install	7	3	1380	4920	3.57
Drop probe	9	1	1560	1260	*0.81*
Advance catheter	8	2	1470	3450	2.35

4 Results

4.1 Preprocessing

The inputs of the analysis are the Petri net containing the process model of the challenge and the event log containing the process executions both related to the PRE and the POST phase. The Petri net could be imported without hassle in PM4Py, but invisible transitions are marked with an INVISIBLE prefix in their label. So, a small change to the import procedure had to be implemented, to make transitions invisible if their label contains the INVISIBLE keyword. Each process execution in the log contains multiple events for each activity: an event with lifecycle Start and an event with lifecycle Complete. Since the event with lifecycle Complete contains all the information that is needed (including the times in which the activity starts and ends, that is the difference between the VIDEOEND and the VIDEOSTART integer attributes), a filtering operation is applied to keep, for each process execution, only the events with lifecycle

Table 2: Effects of deviations on the median throughput time on the Post round log. Median throughput of fit and unfit cases is compared.

Activity	Fit cases	Unfit cases	Fit median throughput time	Unfit median throughput time	Relative throughput
Anesthetize	7	3	1020	960	*0.94*
Put sterile gel	7	3	960	960	1.00
Widen pathway	7	2	960	960	1.00
Puncture	5	5	960	960	1.00
Cover probe	8	2	930	1110	1.19
Check wire in short axis	8	1	960	1440	1.50
Gel in probe	2	8	900	960	1.07
Drap puncture area	9	1	960	1260	1.31
Position probe	8	2	960	960	1.00
Check wire in long axis	1	9	660	960	1.45
Get in sterile clothes	1	9	87300	960	*0.01*
Hand washing	4	6	1080	960	*0.89*
Advance catheter	8	2	930	44370	47.71
Check catheter position	9	1	960	1260	1.31
Check flow and reflow	9	1	960	1260	1.31

Complete. The ROUND attribute is used then to split the original log into a log containing only the process executions Pre training and into a log containing only the process executions Post training.

4.2 Applying Token-based Replay: Throughput Time Analysis

The token-based replay operation could be applied on the Pre log and on the Post log in order to get insights on which process executions contain deviations and to localize them. First, a token-based replay has been performed without localization, and as result each process execution contain at least a deviation. For each case, the fitness value is low, so possibly several transitions are executed in an unfit way according to the process model. A large number of deviations means that the token-based replay might be ineffective in localizing the deviations given the token flooding problem described in the background. The implementation provided by the PM4Py library supports techniques to avoid the token flooding problem (*cleaning_token_flood* parameter of the token-based replay), as described in Section 3. Avoiding the token flooding problem, a localization of the deviations

Table 3: Number of deviations in the PRE phase per resource and activity. The columns are the activities (the corresponding transitions in the model could not fire due to missing tokens). The rows are the resources. Each cell of the inner table is the number of deviations occurred for each couple activity-resource. In the last row, the sum of deviations is reported for each activity. In the last column, the sum of deviations is reported for each resource.

Resource	Advance catheter	Anatomic identification	Anesthetize	Blood return	Check catheter position	Check flow and reflow	Check wire in long axis	Check wire in short axis	Clean puncture area	Compression identification	Cover probe	Doppler identification	Drap puncture area	Drop probe	Gel in probe	Get in sterile clothes	Guidewire install	Hand washing	Position patient	Position probe	Prepare implements	Puncture	Put sterile gel	Remove guidewire	Remove syringe	Remove trocar	Ultrasound configuration	Widen pathway	Wire in good position	**Sum**
R_13_1C	0	3	1	0	0	0	2	1	0	1	1	0	0	0	1	1	0	2	0	0	1	1	0	0	0	1	3	1	0	20
R_14_1D	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	1	1	1	3	1	0	0	0	0	2	1	0	14
R_21_1F	2	1	1	0	0	0	5	4	1	1	1	0	0	0	0	1	1	1	0	3	3	3	0	0	0	1	2	0	0	31
R_31_1G	0	3	2	0	0	0	1	0	1	1	0	1	1	0	1	0	0	1	1	1	5	2	1	0	0	0	2	2	0	26
R_32_1H	0	1	1	1	0	0	2	2	0	1	0	0	0	0	1	1	2	1	0	1	3	0	1	0	0	1	2	1	0	22
R_33_1L	0	1	1	1	0	0	1	1	0	1	0	0	1	0	1	1	0	1	1	0	4	1	1	0	0	1	1	1	0	20
R_45_2A	1	2	0	0	0	0	4	1	0	2	1	0	1	1	0	1	1	1	0	0	2	4	0	0	0	1	1	1	0	25
R_46_2B	0	1	1	0	0	0	1	0	1	1	1	0	0	0	0	1	0	1	1	0	2	0	0	0	0	0	2	1	0	14
R_47_2C	0	1	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	1	0	0	1	0	1	0	0	0	1	1	0	10
R_48_2D	0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	0	1	0	1	2	2	0	0	0	1	2	2	0	18
Sum	3	15	7	2	0	0	17	10	5	11	4	2	3	1	6	9	4	11	4	7	26	14	4	0	0	6	18	11	0	

could be done in a more reliable way. The localization is activated through the *enable_pltr_fitness* parameter of PM4Py. Executing the replay with the option to remove the token flooding and localizing the deviations, different output objects are obtained through the token-based replay:

- A list containing, for each case, the list of transitions that are enabled in the process model.
- Information on places fitness (fit, underfed, overfed executions of the place).
- Information on transitions fitness (fit or unfit executions of the transition). For each unfit execution, the case status is saved (that is a list of attributes along with their last value before the transition is executed). In such way, at the end of the replay it is possible to know how many deviations happened in a particular point of the process, and how many deviations are due to a particular resource.

Thanks to the information on transitions fitness, it is possible to compare an aggregation of the throughput time of fit and unfit process executions. This has been done in tables 1 (comparing the median throughput time of fit and unfit

Table 4: Number of deviations in the POST phase per resource and activity. The columns are the activities. The rows are the resources. Each cell of the inner table is the number of deviations occurred for each couple activity-resource. In the last row, the sum of deviations is reported for each activity. In the last column, the sum of deviations is reported for each resource.

Resource	Advance catheter	Anatomic identification	Anesthetize	Blood return	Check catheter position	Check flow and reflow	Check wire in long axis	Check wire in short axis	Clean puncture area	Compression identification	Cover probe	Doppler identification	Drap puncture area	Drop probe	Gel in probe	Get in sterile clothes	Guidewire install	Hand washing	Position patient	Position probe	Prepare implements	Puncture	Put sterile gel	Remove guidewire	Remove syringe	Remove trocar	Ultrasound configuration	Widen pathway	Wire in good position	**Sum**
R_13_1C	1	1	0	0	0	0	1	0	1	1	0	1	0	0	1	0	0	0	0	0	3	1	0	0	0	0	1	0	0	12
R_14_1D	0	1	1	0	0	0	2	0	1	1	0	1	0	0	1	1	0	1	0	1	4	1	1	0	0	0	1	0	0	18
R_21_1F	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	1	0	0	0	0	3	0	0	0	0	0	1	0	0	11
R_31_1G	0	1	1	0	0	0	1	0	1	0	1	0	0	0	1	1	0	1	0	1	4	0	0	0	0	0	1	0	0	14
R_32_1H	0	2	0	0	1	1	2	0	1	2	1	0	1	0	1	1	0	0	0	0	3	2	1	0	0	0	2	1	0	22
R_33_1L	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	1	0	0	1	0	4	1	1	0	0	0	1	0	0	12
R_45_2A	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	1	0	1	0	0	2	0	0	0	0	0	2	0	0	10
R_46_2B	0	1	1	0	0	0	1	0	1	1	0	0	0	0	1	1	0	1	0	0	2	0	0	0	0	0	2	0	0	12
R_47_2C	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	0	1	1	0	0	0	0	1	1	0	10
R_48_2D	5	1	0	0	0	0	2	2	1	1	0	0	0	0	1	1	0	1	1	0	2	0	0	0	0	0	2	0	0	20
Sum	6	10	3	0	1	1	12	2	10	10	2	3	1	0	8	9	0	6	2	2	28	6	3	0	0	0	14	2	0	

executions in the Pre round log) and 2 (comparing the median throughput time of fit and unfit executions in the Post round log). For most transitions, executing them in an unfit way leads to an higher throughput time. Considering the Pre round, executing in an unfit way the transitions *Position probe*, *Clean puncture area* and *Guidewire install* leads to a throughput time that is, in median, more than 3 times higher in comparison to cases where such transitions were executed according to the process model. This does not hold for some transitions:

- For the Pre round log, the transitions *Widen pathway*, *Cover probe*, *Get in sterile clothes* and *Drop probe* leads to a lower throughput time when executed in an unfit way. For transitions *Widen pathway* and *Get in sterile clothes* nothing could be concluded, since they are executed in a fit way only once, while transition *Drop probe* is executed in an unfit way only once. For the transition *Cover probe*, the difference is more significant, but no reason for this could be pointed out from the authors of this paper.
- For the Post round log, the transitions *Anesthetize*, *Get in sterile clothes* and *Hand washing* report a lower median throughput time when executed in an unfit way. In this case, no statistical significant differences could be identified.

Table 5: Difference between the number of deviations in the POST and in the PRE phase. The columns are the activities. The rows are the resources. Each cell of the inner table is the number of deviations occurred for each couple activity-resource. In the last row, the sum of deviations is reported for each activity. In the last column, the sum of deviations is reported for each resource.

Resource	Advance catheter	Anatomic identification	Anesthetize	Blood return	Check catheter position	Check flow and reflow	Check wire in long axis	Check wire in short axis	Clean puncture area	Compression identification	Cover probe	Doppler identification	Drap puncture area	Drop probe	Gel in probe	Get in sterile clothes	Guidewire install	Hand washing	Position patient	Position probe	Prepare implements	Puncture	Put sterile gel	Remove guidewire	Remove syringe	Remove trocar	Ultrasound configuration	Widen pathway	Wire in good position	**Sum**
R_13_1C	1	-2	-1	0	0	0	-1	-1	1	0	-1	1	0	0	0	-1	0	-2	0	0	2	0	0	0	0	-1	-2	-1	0	-8
R_14_1D	0	0	1	0	0	0	2	0	0	0	0	1	0	0	1	0	0	0	-1	0	1	0	1	0	0	0	-1	-1	0	4
R_21_1F	-2	0	-1	0	0	0	-4	-4	0	0	-1	1	0	0	1	0	-1	-1	0	-3	0	-3	0	0	0	-1	-1	0	0	-20
R_31_1G	0	-2	-1	0	0	0	0	0	0	-1	1	-1	-1	0	0	1	0	0	-1	0	-1	-2	-1	0	0	0	-1	-2	0	-12
R_32_1H	0	1	-1	-1	1	1	0	-2	1	1	1	0	1	0	0	0	-2	-1	0	-1	0	2	0	0	0	-1	0	0	0	0
R_33_1L	0	-1	-1	-1	0	0	0	-1	1	0	0	0	-1	0	-1	0	0	-1	0	0	0	0	0	0	0	-1	0	-1	0	-8
R_45_2A	-1	-1	0	0	0	0	-3	-1	1	-1	-1	0	-1	-1	0	0	-1	0	0	0	0	-4	0	0	0	-1	1	-1	0	-15
R_46_2B	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	1	0	0	0	-1	0	0	0	0	0	0	0	0	-1	0	-2
R_47_2C	0	0	0	0	0	0	0	0	1	0	0	-1	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0
R_48_2D	5	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	-1	0	-2	0	0	0	-1	0	-2	0	2
Sum	3	-5	-4	-2	1	1	-5	-8	5	-1	-2	1	-2	-1	2	0	-4	-5	-2	-5	2	-8	-1	0	0	-6	-4	-9	0	

4.3 Resource Localization of Replay Results

The goal of this section is using the localization information provided by the token-based replay, considering the activity and the resource attribute in order to understand which activities are executed in an unfit way more often, and which resources execute the activities in an unfit way more often.

Tables 3 and 4 contain the output of the token-based replay on the Pre and on the Post log. The columns of both tables are the activities in the log. The rows are the resources. The value in the cell is the number of deviations of the activity performed by given the resource. Summing a row, it is possible to obtain the number of deviations for each resource (so it is possible to discover which resources have problems in executing the process). Summing a column, it is possible to obtain the number of deviations for each activity (so it is possible to discover which activities have fitness problems).

Considering Table 3 (deviations in the Pre log), it is possible to see that two distinct groups (highlighted in Blue and Red) exist among the resources, that have a similar error pattern. A resource (R_47_2C) does not belong to any group. Generally, resources in the Blue group have a larger number of deviations than resources in the Red group.

Table 6: All the durations (calculated as difference between the end time and the start time of the activity) for each activity in the log.

Activity	**Duration values**
Hand washing	5, 5, 6, 6, 7, 12, 12, 12, 13, 15, 17, 19, 19, 21, 21, 27, 32, 36, 44, 51, 72, 104
Ultrasound configuration	6, 7, 7, 8, 9, 9, 10, 15, 15, 16, 19, 19, 19, 23, 24, 28, 29, 35, 37, 39, 43, 44, 45, 47, 47, 49, 53, 53, 57, 58, 59, 102, 120
Anatomic identification	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 15, 22, 31, 44
Compression identification	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 4, 9, 10, 10
Gel in probe	0, 0, 0, 2, 2, 3, 4, 4, 4, 6, 6, 7, 7, 8, 10, 12, 15, 15
Get in sterile clothes	6, 8, 8, 11, 13, 15, 15, 16, 17, 23, 25, 26, 26, 26, 30, 36, 39, 61, 68, 77, 86, 87, 94, 94, 95, 98, 108, 109, 109, 118, 124, 126, 131, 132, 135, 170, 170
Clean puncture area	7, 9, 10, 12, 18, 19, 19, 19, 22, 23, 27, 28, 29, 32, 34, 34, 34, 41, 41
Drap puncture area	20, 21, 24, 28, 29, 30, 31, 32, 34, 34, 34, 34, 36, 36, 37, 38, 39, 41, 51, 58
Cover probe	34, 40, 41, 62, 65, 69, 76, 81, 83, 84, 89, 105, 111, 118, 125, 134, 147, 147, 168, 189
Put sterile gel	0, 3, 4, 4, 4, 6, 6, 6, 7, 7, 7, 8, 8, 8, 9, 9, 12, 18, 22, 23, 26, 30
Prepare implements	0, 0, 5, 6, 7, 8, 10, 13, 13, 17, 17, 18, 19, 20, 21, 21, 23, 23, 23, 23, 23, 24, 24, 26, 27, 27, 28, 29, 30, 30, 30, 33, 36, 38, 40, 43, 44, 45, 49, 50, 50, 52, 58, 62, 65, 67, 69, 79, 81, 84, 84, 86, 86, 91, 93, 94, 94, 95, 104, 118, 120, 126, 143
Anesthetize	2, 3, 4, 5, 5, 5, 6, 6, 6, 7, 8, 8, 8, 11, 12, 12, 18, 19, 34
Puncture	0, 19, 19, 23, 24, 24, 24, 25, 25, 26, 31, 35, 36, 40, 42, 43, 44, 47, 57, 59, 60, 65, 69, 77, 86, 136, 150, 165, 189, 281
Blood return	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 8, 9, 25
Drop probe	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Remove syringe	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 10, 12, 18, 20
Guidewire install	0, 13, 14, 18, 20, 21, 21, 26, 28, 32, 34, 35, 36, 37, 40, 41, 45, 48, 56, 57, 59, 59, 62, 72, 73, 80, 86, 122
Check wire in short axis	0, 0, 0, 0, 0, 2, 3, 3, 4, 5, 5, 5, 6, 7, 7, 9, 9, 10, 10, 11, 14, 14, 14, 17, 20, 26, 37
Check wire in long axis	0, 0, 0, 0, 1, 2, 3, 3, 5, 6, 7, 7, 8, 9, 9, 10, 12, 15, 18, 23, 23, 26, 27, 33, 36, 37, 39, 40, 43, 44, 58
Remove trocar	0, 0, 0, 0, 2, 3, 3, 3, 3, 3, 4, 4, 4, 5, 5, 6, 6, 6, 6, 7, 7, 10, 10, 11
Widen pathway	3, 6, 8, 12, 12, 13, 17, 18, 18, 19, 19, 21, 22, 24, 27, 28, 32, 33, 40, 43, 46
Advance catheter	8, 25, 34, 38, 40, 41, 47, 54, 57, 58, 61, 63, 65, 67, 68, 69, 72, 74, 75, 79, 86, 87, 88, 91, 95, 95, 113, 123, 148, 171
Remove guidewire	0, 0, 0, 0, 2, 2, 3, 3, 4, 5, 6, 9, 9, 14, 17, 20, 24, 26, 33, 37, 38
Check flow and reflow	8, 9, 11, 13, 13, 13, 15, 16, 17, 17, 17, 24, 26, 46, 46, 54, 56, 56, 57
Check catheter position	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 3, 3, 3
Position patient	1, 3, 3, 3, 4, 7
Position probe	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 4, 4, 5, 5, 10, 11, 12, 18
Doppler identification	0, 0, 21, 21, 22
Wire in good position	0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Table 7: Probability distribution (estimated on the durations) for each activity in the log. The log-likelihood of the minimum value of duration and of the maximum value of duration have been reported.

Activity	**Distribution**	**Distribution parameters**	**LL min dur**	**LL max dur**
Hand washing	EXPONENTIAL	0.04	-3.43	-7.34
Ultrasound configuration	EXPONENTIAL	0.03	-3.72	-6.99
Anatomic identification	EXPONENTIAL	0.22	-1.53	-11.09
Compression identification	EXPONENTIAL	0.54	-0.62	-6.00
Gel in probe	UNIFORM	0.00;15.00	-2.71	-2.71
Get in sterile clothes	UNIFORM	6.00;164.00	-5.10	-5.10
Clean puncture area	UNIFORM	7.00;34.00	-3.53	-3.53
Drap puncture area	NORMAL	34.35;8.73	-4.44	-6.75
Cover probe	UNIFORM	34.00;155.00	-5.04	-5.04
Put sterile gel	EXPONENTIAL	0.10	-2.33	-5.24
Prepare implements	EXPONENTIAL	0.02	-3.85	-6.89
Anesthetize	EXPONENTIAL	0.11	-2.46	-5.85
Puncture	EXPONENTIAL	0.02	-4.16	-8.55
Blood return	EXPONENTIAL	0.60	-0.50	-15.61
Drop probe	IMMEDIATE			
Remove syringe	EXPONENTIAL	0.39	-0.93	-8.81
Guidewire install	NORMAL	44.11;26.06	-5.61	-8.64
Check wire in short axis	EXPONENTIAL	0.11	-2.18	-6.37
Check wire in long axis	EXPONENTIAL	0.06	-2.86	-6.17
Remove trocar	UNIFORM	0.00;11.00	-2.40	-2.40
Widen pathway	UNIFORM	3.00;43.00	-3.76	-3.76
Advance catheter	NORMAL	73.07;33.93	-6.28	-8.61
Remove guidewire	EXPONENTIAL	0.08	-2.48	-5.65
Check flow and reflow	UNIFORM	8.00;49.00	-3.89	-3.89
Check catheter position	EXPONENTIAL	1.13	0.13	-3.27
Position patient	UNIFORM	1.00;6.00	-1.79	-1.79
Position probe	EXPONENTIAL	0.35	-1.06	-7.31
Doppler identification	UNIFORM	0.00;22.00	-3.09	-3.09
Wire in good position	IMMEDIATE			

Table 8: Number of outliers (having log-likelihood < -8 according to the discovered distribution of executions times for an activity). The columns are the activities. The rows are the resources. Each cell of the inner table is the number of outliers occurred for the couple activity-resource. In the last row, the number of outliers is reported for each activity. In the last column, the sum of outliers is reported for each resource.

Resource	Advance catheter	Anatomic identification	Anesthetize	Blood return	Check catheter position	Check flow and reflow	Check wire in long axis	Check wire in short axis	Clean puncture area	Compression identification	Cover probe	Doppler identification	Drap puncture area	Drop probe	Gel in probe	Get in sterile clothes	Guidewire install	Hand washing	Position patient	Position probe	Prepare implements	Puncture	Put sterile gel	Remove guidewire	Remove syringe	Remove trocar	Ultrasound configuration	Widen pathway	Wire in good position	**Sum**
R_13_1C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R_14_1D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R_21_1F	2	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	3	3	3	0	0	0	1	0	0	0	17
R_31_1G	0	3	2	0	0	0	1	0	1	1	0	1	1	0	1	0	0	1	1	1	5	2	1	0	0	0	2	2	0	26
R_32_1H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R_33_1L	0	1	1	1	0	0	0	1	0	1	0	0	1	0	1	1	0	0	1	0	4	1	0	0	0	1	1	0	0	16
R_45_2A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R_46_2B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R_47_2C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R_48_2D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sum	2	5	3	1	0	0	1	1	1	3	1	1	2	0	2	2	1	1	2	4	12	6	1	0	0	2	3	2	0	

Table 5 reports the difference between the number of deviations in the Pre log and in the Post log (the lower is the number, the bigger is the improvement in the correctness of the process execution). It is interesting to note that the Blue group reports bigger improvements in comparison to the Red group.

Tables 3, 4 and 5 are useful not only from an instructor's perspective (being able to identify common error patterns thank to the tables) but also from the student's perspective (to know which activities are performed by them in an uncorrect way). An important remark is that these results are obtained without any sort of noise cleaning. The description of the challenge asserts that noise might be contained in the log. In the following sections, an outlier removal technique based on learning, for each activity, a probability distribution that describes the execution times, will be presented.

4.4 Learning a Probability Distribution for the Time Performance

The goal of this section is to consider the execution times of the activities and try to infer a probability distribution that fits the data.

All the execution times, for each activity in the log, are reported in Table 6. It is interesting to observe that:

Table 9: Number of deviations in the PRE phase per resource and activity, after the outlier removal has been performed. The columns are the activities. The rows are the resources. Each cell of the inner table is the number of deviations occurred for the couple activity-resource. In the last row, the sum of deviations is reported for each activity. In the last column, the sum of deviations is reported for each resource.

Resource	Anatomic identification	Anesthetize	Blood return	Check catheter position	Check flow and reflow	Check wire in long axis	Check wire in short axis	Clean puncture area	Compression identification	Doppler identification	Drap puncture area	Drop probe	Gel in probe	Guidewire install	Hand washing	Position patient	Position probe	Prepare implements	Puncture	Put sterile gel	Remove guidewire	Remove syringe	Remove trocar	Ultrasound configuration	Widen pathway	Wire in good position	**Sum**
R_13_1C	3	1	1	1	1	1	2	1	1	0	0	1	1	0	2	0	0	0	0	1	1	1	1	1	1	0	22
R_14_1D	0	1	1	1	1	0	0	1	0	0	0	1	1	0	1	1	2	0	0	0	1	0	1	0	1	0	14
R_21_1F	0	0	4	0	1	3	4	1	1	0	1	4	1	0	0	0	4	0	0	1	2	2	2	1	1	1	34
R_31_1G	2	2	2	0	1	0	1	1	0	1	1	2	1	0	0	1	2	0	0	2	1	2	2	0	2	0	26
R_32_1H	1	1	1	1	1	1	3	1	1	0	1	1	1	0	0	0	2	0	0	1	0	0	1	1	1	0	20
R_33_1L	0	1	2	0	0	0	1	1	1	0	1	2	1	0	0	1	0	0	0	2	1	2	1	0	1	0	18
R_45_2A	2	0	1	1	1	3	1	0	2	0	1	2	0	1	0	0	1	0	0	1	1	2	1	0	1	1	23
R_46_2B	1	0	1	1	1	0	1	1	1	0	0	1	1	0	1	1	1	0	0	1	1	1	1	1	1	0	18
R_47_2C	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	0	1	0	0	1	1	1	1	0	1	0	17
R_48_2D	1	0	2	1	1	1	1	1	1	0	1	2	1	0	0	0	1	0	0	0	1	2	2	1	2	0	22
Sum	11	7	16	7	9	9	14	9	9	2	7	17	9	1	4	4	14	0	0	10	10	13	13	5	12	2	

- The activities *Drop probe* and *Wire in good position* are always immediate (the difference between the VIDEOEND and the VIDEOSTART attribute is always 0).
- Some activities (*Anatomic identification*, *Compression identification*, *Blood return*, *Remove syringe*, *Position probe*) are sometimes immediate, sometimes they require some seconds.

For each activity, a probability distribution that describes the execution times could be obtained. For each activity that has at least one value greater than 0, the following probability distributions are considered:

- Uniform (in an interval): the parameters (e.g. the infimum and the maximum of the interval) are the minimum and the maximum value of the interval.
- Normal, with parameters μ (the mean) and σ (the standard deviation).
- Exponential, with parameter λ (the mean of this distribution is $\frac{1}{\lambda}$).

The log-likelihood is tested against each probability distribution, to choose which one fits better the data. Table 7 contains for each activity the distribution (along with the parameters) that better fits the execution times in the log for the activity. Also, the log-likelihood of the minimum and the maximum value contained in the execution times for the given activity are reported in Table 7.

Table 10: Number of deviations in the POST phase per resource and activity, after the outlier removal has been performed. The columns are the activities. The rows are the resources. Each cell of the inner table is the number of deviations occurred for the couple activity-resource. In the last row, the sum of deviations is reported for each activity. In the last column, the sum of deviations is reported for each resource.

Resource	Anatomic identification	Anesthetize	Blood return	Check catheter position	Check flow and reflow	Check wire in long axis	Check wire in short axis	Clean puncture area	Compression identification	Doppler identification	Drap puncture area	Drop probe	Gel in probe	Guidewire install	Hand washing	Position patient	Position probe	Prepare implements	Puncture	Put sterile gel	Remove guidewire	Remove syringe	Remove trocar	Ultrasound configuration	Widen pathway	Wire in good position	**Sum**
R_13_1C	1	1	0	1	1	1	1	1	0	1	1	1	1	0	1	0	0	0	0	1	1	1	1	1	0	1	18
R_14_1D	1	1	1	1	1	2	1	1	1	1	1	1	1	0	1	0	0	0	0	0	1	0	1	0	1	1	19
R_21_1F	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	1	0	1	1	0	1	1	19
R_31_1G	1	1	1	1	1	1	0	1	0	0	1	1	1	0	0	0	2	0	0	0	1	1	1	0	1	1	17
R_32_1H	2	1	2	1	1	1	2	1	2	0	1	2	1	0	1	0	0	0	1	1	0	2	2	1	1	0	26
R_33_1L	0	1	1	1	1	0	0	1	1	0	1	1	1	0	1	1	1	0	0	1	1	1	1	1	1	1	19
R_45_2A	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	0	1	1	18
R_46_2B	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	1	1	0	1	1	20
R_47_2C	1	1	1	1	1	1	0	1	1	0	0	1	1	0	1	0	1	0	0	1	0	1	1	0	1	0	16
R_48_2D	1	0	1	1	1	1	2	1	1	0	1	1	1	0	1	1	1	0	0	0	0	1	1	1	1	1	20
Sum	10	9	10	10	10	10	9	10	9	3	9	11	10	0	9	2	7	0	1	7	5	9	11	4	9	8	

The more the log-likelihood is low, the more is likely that the considered value is an outlier for the activity. At first glance, identifying outliers, and which activities/resources have more outliers, is important to identify the activities that need more focus by the instructor, and which resources takes too little or too much time to execute the activity. Table 8 reports for each activity and resource the number of executions that were considered as outliers (given the execution time). The log-likelihood threshold that is being considered is -8: every execution time that has a log-likelihood smaller than -8 is considered as outlier. The columns are the activities in the log. The rows are the resources. The value in the cell is the number of times for which the execution time is an outlier. Summing a row, it is possible to obtain the number of outliers for each resource (so it is possible to discover which resources are too fast / too slow in executing the process). Summing a column, it is possible to obtain the number of outliers for the given activity (so it is possible to discover which activities are more prone to outliers).

Another outlier detection technique, that could be deployed for this purpose, is the box plot. This is a method for graphically depicting groups of numerical

Table 11: Difference between the number of deviations in the POST and in the PRE phase, after the outlier removal has been performed. The columns are the activities. The rows are the resources. Each cell of the inner table is the number of deviations occurred for the couple activity-resource. In the last row, the sum of deviations is reported for each activity. In the last column, the sum of deviations is reported for each resource.

Resource	Anatomic identification	Anesthetize	Blood return	Check catheter position	Check flow and reflow	Check wire in long axis	Check wire in short axis	Clean puncture area	Compression identification	Doppler identification	Drap puncture area	Drop probe	Gel in probe	Guidewire install	Hand washing	Position patient	Position probe	Prepare implements	Puncture	Put sterile gel	Remove guidewire	Remove syringe	Remove trocar	Ultrasound configuration	Widen pathway	Wire in good position	**Sum**
R_13_1C	-2	0	-1	0	0	0	-1	0	-1	1	1	0	0	0	-1	0	0	0	0	0	0	0	0	0	-1	1	-4
R_14_1D	1	0	0	0	0	2	1	0	1	1	1	0	0	0	0	-1	-2	0	0	0	0	0	0	0	0	1	5
R_21_1F	1	1	-3	1	0	-2	-3	0	0	1	0	-3	0	0	1	0	-4	0	0	0	-2	-1	-1	-1	0	0	-15
R_31_1G	-1	-1	-1	1	0	1	-1	0	0	-1	0	-1	0	0	0	-1	0	0	0	-2	0	-1	-1	0	-1	1	-9
R_32_1H	1	0	1	0	0	0	-1	0	1	0	0	1	0	0	1	0	-2	0	1	0	0	2	1	0	0	0	6
R_33_1L	0	0	-1	1	1	0	-1	0	0	0	0	-1	0	0	1	0	1	0	0	-1	0	-1	0	1	0	1	1
R_45_2A	-1	1	0	0	0	-2	0	1	-1	0	0	-1	1	-1	1	0	0	0	0	0	-1	-2	0	0	0	0	-5
R_46_2B	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	-1	0	0	0	0	0	0	0	-1	0	1	2
R_47_2C	0	0	0	0	0	1	0	0	0	-1	-1	0	0	0	1	0	0	0	0	0	-1	0	0	0	0	0	-1
R_48_2D	0	0	-1	0	0	0	1	0	0	0	0	-1	0	0	1	1	0	0	0	0	-1	-1	-1	0	-1	1	-2
Sum	-1	2	-6	3	1	1	-5	1	0	1	2	-6	1	-1	5	-2	-7	0	1	-3	-5	-4	-2	-1	-3	6	

data through their quartiles. Box plots may also have lines extending vertically from the boxes indicating variability outside the upper and lower quartiles.

Although, also these results shall be taken with a pinch of salt: in the description of the challenge, indeed, it is assumed that the log contains noise. So, an activity may take a lot of time because the student is slow in performing it, or because it is actually noise recorded in the event log.

4.5 Replay after Outliers Removal

The goal of this section is to repeat the experiments described in Section 4.3 on the resource and the activity localization of the replay results, but removing from the log any event that could be considered as outlier (having log-likelihood threshold lower than -8 with regards to the probability distribution that better fits the execution time for the given activity). In such way, we can avoid the activation of parts of model that did not happen. Although, the risk is to not activate parts of the model that did actually happen, but were executed in a too fast/too slow way by the student. Anyhow, a comparison between the results of

the localization with noise removal and the results of the localization without noise removal could help to understand whether deviations as described in tables 3 and 4 are really meaningful or were possibly due to noise in the model.

The results of the localization after the removal of the noise are reported in tables 9, 10. The difference between these two tables is then reported in Table 11. It is interesting to see how deviations signaled in Section 4.3 for activities *Prepare implements* and *Puncture* completely disappear in the new tables. This could be related to two different reasons:

- Some events related to *Prepare implements* and *Puncture* activities, reported as outliers in Table 8, are noise.
- Some events before *Prepare implements* and *Puncture* activities were noisy, and the procedure to avoid the token flooding removed useful tokens to activate the corresponding transitions.

In any case, deviations on *Prepare implements* and *Puncture*, that were seemingly meaningful in tables 3 and 4, were mostly due to noise in the log.

5 Conclusion

In this report, that competes for the Academic category of the Conformance Checking Challenge 2019, and is targeting both the student's and the instructor's perspectives, two questions have been considered:

1. Which are the activities that are more prone to errors? Which resources are more prone to errors, i.e. to execute in a wrong way the process? Are there groups (clusters) of resources that share the same errors?
2. Is it possible to learn, for each activity, a time-performance distribution that is able to tell if a given execution is fit or unfit according to the execution times of the same activity in other process executions?

The conformance checking operations were performed using the PM4Py process mining library, that implements a token-based replay providing localized outputs. These could be used to perform comparative throughput analysis between the cases where a transition has been executed in a fit way, and the cases where a transition has been executed in an unfit way. Moreover, the library implements the discovery of a probability distribution that is associated to the execution times of an activity.

For the question (1), the tables 3 (Pre round), 4 (Post round) have been produced applying the localized token-based replay on a filtered log containing only the Pre and the Post round respectively. In these tables, each row is corresponding to a resource, and each column is corresponding to an activity. It is easy to understand which resources and activities are more problematic. Moreover, it is possible to identify two similar groups of resources, that show a similar error pattern. These tables are useful both on the instructor's side (since they permit to understand which activities are more problematic and to discover groups of

resources with a similar error pattern) and on the student's side (since the errors performed by each student in the Pre phase are clearly reported). Moreover, Table 5 permits to understand how the two groups improved in the process execution after the practice (the Blue group, that was initially more problematic, improves a lot in comparison to the Red group).

The transition-specific reports on the effects of deviations on the overall throughput time are reported in tables 1 and 2. In most cases, executing a transition in an unfit way leads to an higher throughput time for the process: considering the Pre round, executing in an unfit way the transitions *Position probe*, *Clean puncture area* and *Guidewire install* leads to a throughput time that is, in median, more than 3 times higher in comparison to cases where such transitions were executed according to the process model.

To produce tables 3, 4 and 5, noise removal on the log has not been performed. Since the challenge description clearly states that the log contains noise, this means that the results might be affected by noise. To perform noise removal, a technique based on the likelihood of the execution time of an activity (in comparison to the other execution times) has been deployed.

To evaluate the likelihood of the execution times, a probability distribution has been estimated out of the execution times (a normal, an exponential and an uniform distribution have been considered, and the best fit according to the log-likelihood has been taken). Values that are unlikely according to the probability distribution have low log-likelihood, this helps in identifying slow students or activities that require more standardization (question (2)). In Table 8, the number of outliers (log-likelihood < 8) for each activity and resource has been reported (each row is corresponding to a resource, and each column is corresponding to an activity). Although, also this analysis may be affected by the noise in the log. Assuming some noise in the log, it is possible that the activity takes more or less time than expected because it is actually noise in the log.

In Section 4.5, the experiments on resource and activity localization of token-based replay are repeated removing the outliers, reported in Table 8, to see which results reported in the tables 3 and 4 were affected by noise. Tables 9, 10 and 11 were created after the application of the token-based replay on the cleaned log. For activities *Prepare implements* and *Puncture*, deviations completely disappear after removing the noise. So, activities *Prepare implements* and *Puncture* are executed correctly according to the process model, and the signaled deviations were due to the noise.

Overall, this reports aims to provide conformance checking information that is useful both on the instructor's and the student's side, and aims to make light on the parts of the process model that are executed in an unfit way, and which activities could be target of improvements (because the execution time for some students is too high).

References

1. van der Aalst, W.: Structural characterizations of sound workflow nets. Computing Science Reports **96**(23), 18–22 (1996)
2. van der Aalst, W., Adriansyah, A., De Medeiros, A.K.A., Arcieri, F., Baier, T., Blickle, T., Bose, J.C., Van Den Brand, P., Brandtjen, R., Buijs, J., et al.: Process mining manifesto. In: International Conference on Business Process Management. pp. 169–194. Springer (2011)
3. Adriansyah, A.: Aligning observed and modeled behavior (2014)
4. Adriansyah, A., Sidorova, N., van Dongen, B.: Cost-based fitness in conformance checking. In: Application of Concurrency to System Design (ACSD), 2011 11th International Conference on. pp. 57–66. IEEE (2011)
5. Carmona, J., Dongen, B., Solti, A., Weidlich, M.: Conformance Checking: Relating Processes and Models. Springer (2018)
6. De Leoni, M., van der Aalst, W., Dees, M.: A general framework for correlating business process characteristics. In: International Conference on Business Process Management. pp. 250–266. Springer (2014)
7. Kerremans, M.: Gartner Market Guide for Process Mining, Research Note G00353970 (2018), www.gartner.com
8. Leemans, S., Fahland, D., van der Aalst, W.: Scalable process discovery and conformance checking. Software & Systems Modeling **17**(2), 599–631 (2018)
9. Martínez, J., Silva, M.: A simple and fast algorithm to obtain all invariants of a generalised Petri net. In: Application and Theory of Petri nets, pp. 301–310. Springer (1982)
10. Miyamoto, T., Kumagai, S.: Calculating place capacity for Petri nets using unfoldings. In: Application of Concurrency to System Design, 1998. Proceedings., 1998 International Conference on. pp. 143–151. IEEE (1998)
11. Munoz-Gama, J., de la Fuente, R., Sepulveda, M., Fuentes, R.: Conformance Checking Challenge 2019 (CCC19). https://doi.org/10.4121/uuid:c923af09-ce93-44c3-ace0-c5508cf103ad. https://doi.org/10.4121/c923af09-ce93-44c3-ace0-c5508cf103ad
12. Ramezani, E., Fahland, D., van der Aalst, W.: Where did i misbehave? diagnostic information in compliance checking. In: International conference on business process management. pp. 262–278. Springer (2012)
13. Robitaille, D.: Root cause analysis: basic tools and techniques. Paton Professional (2004)
14. Rogge-Solti, A., Senderovich, A., Weidlich, M., Mendling, J., Gal, A.: In log and model we trust? a generalized conformance checking framework. In: International Conference on Business Process Management. pp. 179–196. Springer (2016)
15. Rozinat, A., van der Aalst, W.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1), 64–95 (2008)