Information and Software Technology 80 (2016) 110–123

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

# Analyzing and repairing overlapping work items in process logs

Ahmed Awad[a,*], Nesma M. Zaki[a], Chiara Di Francescomarino[b]

[a] Faculty of Computers and Information, Cairo University, Giza, Egypt
[b] Fondazionre Bruno Kessler, Trento, Italy

ARTICLE INFO

*Article history:*
Received 21 February 2016
Revised 1 August 2016
Accepted 30 August 2016
Available online 31 August 2016

*Keywords:*
Process logs
Humans work patterns
Performance measurement
Business processes
Log analysis

ABSTRACT

**Context:** In real life logs, it often happens that some human resources appear to have more than one task active concurrently, thus resulting in human multitasking. However, tasks that require some intellectual effort cannot be executed in parallel in real life. This misalignment between what actually happens and what is registered in the logs, however, is not reflected in the output of the different log-based performance measuring approaches, thus compromising the quality of the computed metrics.

**Objective:** We introduce a novel approach to rewrite events in process execution logs for multitasking human resources. The approach is based on two typical human work patterns, the queuing and stacking patterns. The rewrite aims at *serializing* multi tasks for the same resource based on the work pattern detected. Thus, possibly better performance measures can be obtained.

**Method:** We defined a quantitative approach to detect multitasking human performers and resolve them by serialization. The approach is prototyped and evaluated on a set of real-life software development process logs.

**Results:** Our results show that the proposed approach contributes to find better results when log-based performance analysis techniques are applied to the repaired logs in comparison to the original logs.

**Conclusions:** The work shows that based on the human work patterns, stacking or queuing, logs can be enhanced, so as to be possibly closer to what happened in the reality and to allow for more accurate performance measures.



## 1. Introduction

Processes, e.g., software processes or companies' business processes, are nowadays always more supported by tools aiming at facilitating their execution. These instruments, as for instance project and task management tools in software development processes, besides supporting the process enactment, are also able to keep track of the process execution in log files. An execution log carries all details about the work items that have been executed, as their timestamp, human performers and data that have been processed. The work items can be traced in the log at different levels of detail, e.g., the occurrence of the event corresponding to the whole work item can be tracked, or the different phases of the lifecycle of the work item (e.g., the *start* and the *end* events of the work item) can be monitored. Such information contained in execution logs is valuable and can be used for measuring the process performance. The quality of the log depends on the level of maturity and automation of the process execution within the organization [21].

By observing execution logs of human-intensive processes, it comes out that some human resources may appear to work on more than one work item simultaneously. That is, the *start* events for two or more distinct work items are observed in succession for the same human performer. Yet, no *complete* event is observed for the first work item before observing the start event of the next work item. Thus, this human performer appears to be *multitasking*. There are some studies that show that humans can be trained to multitasking [20]. However, this is effective for simple work items like answering a phone call while writing an email. Work items that require more intellectual effort, e.g., writing a computer program or evaluating an insurance claim, cannot be effectively performed concurrently with other work items, especially if these work items are of the same type. Thus, we can claim that a human resource is effectively working on only one of the concurrent work items whereas all other work items are *waiting* to be processed, either because they have been queued (*queuing* pattern) or because they have been interrupted randomly or due to a human decision (*stacking* pattern) [5]. Workers' attitude towards incoming

* Corresponding author.
*E-mail addresses:* a.gaafar@fci-cu.edu.eg (A. Awad), n.mostafa@fci-cu.edu.eg (N.M. Zaki), dfmchiara@fbk.eu (C.D. Francescomarino).

http://dx.doi.org/10.1016/j.infsof.2016.08.010
0950-5849/

work items in real working environments, indeed, is commonly defined as queue attitude when workers prefer to queue new incoming work items, and as stack attitude, when workers prefer to quickly complete the incoming work items.[1]

From process execution logs, we are able to measure Performance Indicators [19,21,27]. For example, workload for all resources within a time period and cycle time are among performance indicators that can be extracted from an execution log. Most of the current approaches that measure human performance based on logs [3,13,32] do not account for the "apparent" *multitasking* of human performers when extracting measures, thus affecting the accuracy of the performance measures.

In this paper, we introduce a novel approach to rewrite events in a process execution log for addressing the accuracy loss due to the "apparent" *multitasking* of human performers in logs tracing human work items. The proposed approach is based on the evidence that at a specific time a human resource can only be working actively on one work item - other work items are actually *waiting* to be processed - and on two human work patterns. Inline with the real-world observations about human attitude towards incoming work items, the two work patterns are the *queuing* and the *stacking* work patterns. According to the former, all new work items activated by the resource are actually queued waiting for being processed. In the latter approach, newly activated work items have higher priority than older ones and they are put in focus of the performer and the older is *suspended*. When these patterns are detected in an execution log, the log can be rewritten, by introducing new events or replacing erroneous ones, in order to make the log explicitly reflect the pattern and consistent with the real world. The enhanced log can then be provided as input to the different performance measuring approaches in order to obtain more accurate results.

In summary, the contributions of this paper are the following ones:

- A formalization of the notion of overlapping work items based on the queuing and stacking work patterns has been proposed;
- different overlap-resolution strategies and their applicability to the detected work pattern have been proposed and discussed;
- a strategy to resolve multiple overlapping work items, called compound overlaps, has been presented;
- a prototypical implementation of the proposed approach and an evaluation based on real-life software process logs has been performed.

The rest of this paper is organized as follows: Section 2 briefly discusses some of the background concepts and techniques that are used throughout the paper. Section 3 and Section 4 present the problem we are going to face and our contribution in enhancing process execution logs, respectively. In Section 5, we evaluate the proposed approach on real life logs. Related work is discussed in Section 6. A critical discussion of the approach and an outlook on future work (Section 7) conclude the paper.

## 2. Background

In this section, we describe some of the background concepts. When processes are enacted several process instances, cases, are generated based on a process definition. A case evolves by the evolution of its work items. Work items evolve according to a predefined state machine called the work item life cycle cf. Section 2.1. Work item evolution is reflected in an event that is generated by the execution environment carrying all data about the work item state change. The events from the different running instances are stored in so-called execution logs cf. Section 2.2. The details contained in the log depends on several factors like the execution environment and the awareness of the human performers to log their work items as detailed as possible. These logs are then used for different types of analysis. Among those are performance measurement techniques [14,26,32]. These techniques quantitatively measure a predefined set of metrics cf. Section 2.3. In this paper, we use this performance measuring techniques on the log before and after the repair. Section 2.4 briefly discusses interval algebra as it will be needed to formally ground work item overlapping relations.

### 2.1. Work item life cycle

Fig. 1 describes the reference work item life cycle for human activities adapted from [25] by refraining from automatic work items as we assume that their impact on the overall case performance is negligible. We describe here only the states and transitions that are relevant for this work. For more details, the reader is referred to [25]. Once a work item is *created*, the system offers it to one or more resources (the work item is in the *offered* state) or allocates the work item directly to one of the resources (the work item is in the *allocated* state). An example, on the *offered* state is the case when a new development sprint starts and all work items are made available for the team.

An *offered* work item can be picked by a resource. The work item gets *allocated* to that resource. Next, a resource can *start* working on that item. Normally, the resource can declare the work item *completed* once he has done work with it. However, the resource may *suspend* the work item and later on resumes it. A resource can declare the work item as *failed* if she was unable to complete that work. In this case, the work item needs to be redone.

All these state transitions are represented as entries in the process execution log, i.e., as *execution events*. Definition 2.1 formalizes the notion of an execution event.

**Definition 2.1** [Execution Event]**.** Let PM be the set of all process models, PI be the set of all process instances and AI be the set of all work items, and R be the set of all resources. An execution event is a tuple (state, workItem, processInstance, timestamp, resource), where:

- $state \in \{created, offered, allocated, started, suspended, completed, failed\}$ to indicate the actual state of the event,
- $workItem \in AI \cup \{\perp\}$ is a reference to the activity instance (work item) for which the event occurred,
- $processInstance \in PI$ is a reference to the process instance within which the event occurred,
- $timestamp \in \mathbb{N}$ indicates the time at which the event occurred,
- $resource \in R \cup \{\perp\}$ is a reference to the human resource performing a work item. When this property is not applicable, e.g., this is an automated step, this property has the value $\perp$.

For instance the tuple $(started, 2142, 215, 2014-11-308:13,554)$ is the execution event corresponding to the start state of work item 2142 of the process instance (case) 215 executed by resource 554 on 2014-11-30.

### 2.2. Execution log

An execution log stores the footprint of a process execution. Mathematically, a log is seen as a sequence of events that are partially ordered by their timestamps. Each event represents a state transition of one work item within a process model. Table 1 shows a snapshot of an execution log.[2]

[1] http://archive.is/Ik9ru.

[2] For the sake of readability we report here the timestamp in the date format.