BACHELOR

A Method to determine Actual Time Worked from Event Logs

Roest, Ryan

Award date:
2022

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Bachelors End Project
Postbus 513, 5600 MB Eindhoven
The Netherlands
www.tue.nl

Author
R. Roest (S158607)

Date
April 26, 2022

A Method to determine Actual Time Worked from Event Logs

R. Roest (S158607)
r.roest@student.tue.nl

Where innovation starts

Abstract

This paper describes two methods for the calculation of actual time worked. This statistic is useful for increasing visibility and overview of long processes, and distinguishing elapsed time between performing work and waiting for customer response. To develop these methods, the CRISP-DM framework is used. A own dataset is created to verify the calculations. Our results show that the proposed methods, the accumulation and theoretical approach, lead to a upper and a lower bound of the ground truth respectively. Furthermore, the actual time worked is used to determine user profile types, based on consistency and the amount of hours worked per week. Additional work is needed to perform consistent estimation of the actual time worked, and the determination of user profile types can be automated.

Keywords:process mining, profile-type, throughput time, actual time worked

Technische Universiteit
Eindhoven
University of Technology

Table of contents

Title
A Method to determine Actual Time Worked from Event Logs

1 Introduction

The ever-increasing data collection leads to more possibilities of gaining insight into large processes. Customer responses introduce large amounts of variability in throughput times of processes, which calls for a different approach to better encapsulate the actual time spent on the process. Employers need better insights and a clearer overview of large processes in order to address employees properly. Effective performance management can lead to an increase in productivity of employees (Tomczak, Lanzo, & Aguinis, 2018), thus actual time worked calculation can be a valuable tool to achieve this goal. "Sustained efficient and effective employee performance requires the manager's ongoing attention and involvement."(McConnell, 2004)

Time Management is effective to increase time spent on high important tasks (Hall & Hursch, 1982). When time assessment is not possible, due to lack of visibility of actual time worked, then there is no option to improve the effectiveness of struggling employees. It is imperative for employers to have visibility on these important data points since struggling employees do not perform optimally. When visibility is increased, managers can more easily identify bottlenecks in the employees' performance and thus improve the effectiveness of their employees. When we are working at home, it is harder to provide effective counseling for employees. All the more reason to improve visibility of employee time worked and check long-standing projects. The current method to measure this time is the throughput time. This is nothing more than the subtraction of the starting timestamp from the ending timestamp of a trace. Throughput time is a very one-dimensional measure of the actual time worked, which does not take into account the amount of work put in by employees and the time spent waiting on the replies of customers. To include these aspects a different approach of calculating a more true to reality statistic is needed, which leads to the following research question:

How can the actual time worked by employees be determined from Event logs?

To answer this question, multiple parts of this larger question need to be tackled separately. Differences in data sets generate a multitude of nuances, which lead to the following questions. First, the type of tasks employees performs require a different way of attributing the amount of time spent on work. Since one user can be assigned to multiple traces, it is important to keep track of task switches. These track switches are hereby referred to as parallel task processing. Time between a task switch is hard to allocate. It can be spent on break, on the previous task or on the next task. This problem can be tackled in two distinct ways, leading to the development of two different approaches. One method is uses the structure of the data, to find defined starting and ending points on time spent working. This method is hereby referred to as the theoretical method. The other relies on the allocation of time of users were present, which will be referred to by the accumulation method. To give insight into these two methods and their effect on the calculation, sub-question 1.1 is formulated as: "What is the difference between the theoretical and accumulation methods of actual time worked calculation?"

Secondly, with the creation of a new statistic, verification methods need to be developed to prove its correctness. While most mathematical papers rely on proofs to achieve this, the real-life data set properties make this hard to accomplish. By calculating actual time worked, it will rely on the trial & error and reasoning for its validity. This leads to sub-question 1.2, "How can the actual time worked calculation be verified?".

Lastly, another point of interest is the distinction of type of users present in the dataset. Calculating actual time worked leads to a bigger insight into the behaviour of users. Following this behaviour and recognition of patterns show a difference in the type of tasks users perform. This will have an effect on the results of the two methods.

2 Methodology Crisp-DM

The framework used for actual time worked calculation is the CRISP-DM, the CRoss Industry Standard Process for Data Mining. This framework is a standardized process to conduct data, and process, mining. It streamlines the process and breaks the work up into six steps, as seen in figure 2.1 (Wirth & Hipp, 2000). This chapter describes each step in the CRISP-DM framework and explains they are used to develop a method for calculating the actual time worked.

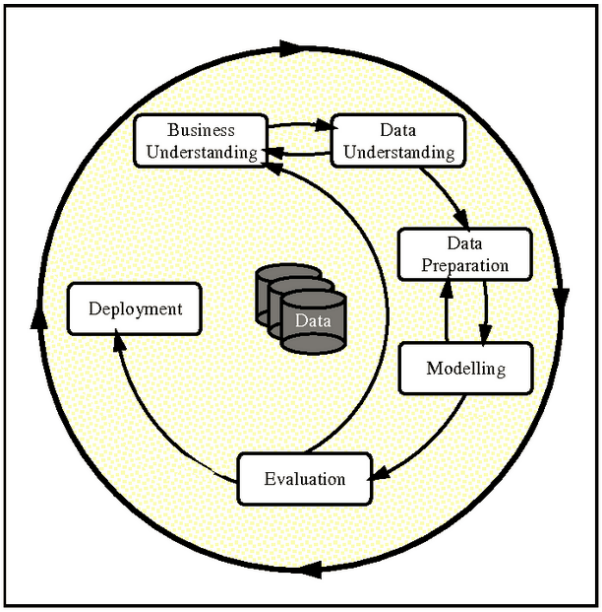

Figure 2.1: Phases of CRISP-DM (Wirth & Hipp, 2000)

2.1 Business Understanding

The first step of the Crisp-DM framework consists of business understanding. The goals are to understand the objectives of the business and set out a preliminary project plan to achieve these goals. The business in question is a Dutch financial institute, which processes loan applications among other things. These applications consist of multiple events, which differ in type. Some events require the response of the customer, which introduces a large amount of variability in the completion times of the applications. The goal of this paper is relevant for the business, as the insights provided by a correct actual time worked calculation can lead to improvement in the performance of employees. In addition, it provides the managers with a clearer view of the applications, and employee performance. Lastly, large gaps between actual time worked and throughput time can indicate waiting on responses from customers. To increase the business and subsequently the data understanding of the data set, the paper of Moreira is used concerning the BPI Challenge 2017. The participants submitted papers for this challenge, with many including data exploration steps. The paper explains the different application states and provides a high-level overview of the process. It attempts to analyze the process to improve customer experience by streamlining. With the use of predictive analysis and the introduction of calculated variables, they produce accurate predictions and show which steps are indicative of the success rate of an application (Moreira, Haven, Sozzo, & Wichert, 2018).

2.2 Data Understanding

Given that the data used is from a real-life company, it is bound to be noisy data. The amount of noise and unfinished traces can lead to confusing and misleading business process models. Weijters and Ribeiro sketch a situation where low-structured data and noise leads to 'spaghetti' models. To generate easy to understand process models, split/join frequency tables are used (Weijters & Ribeiro, 2011).

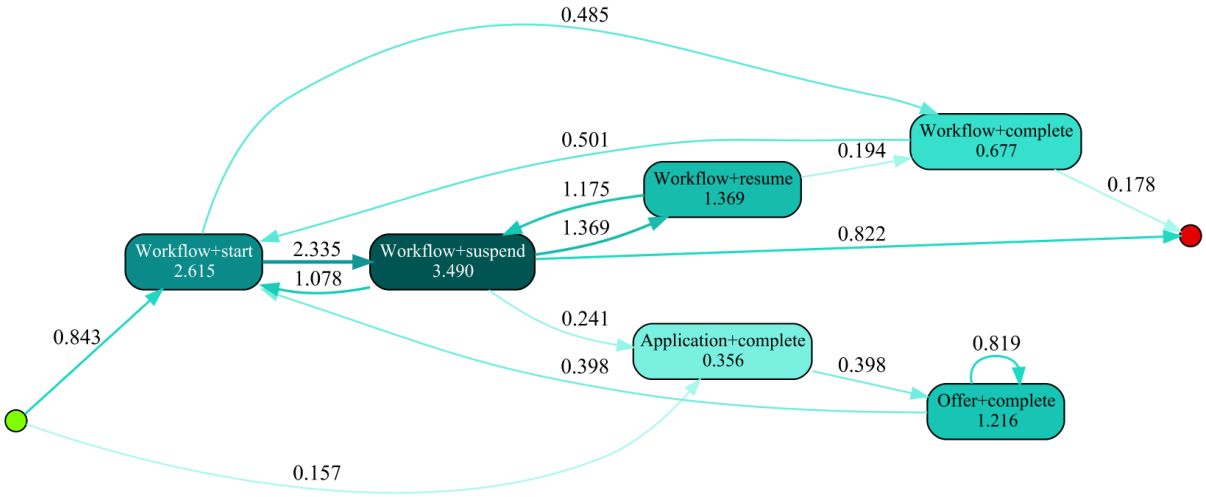

Figure 2.2: Petrinet Life-cycle Transitions

Technische Universiteit **Eindhoven** University of Technology

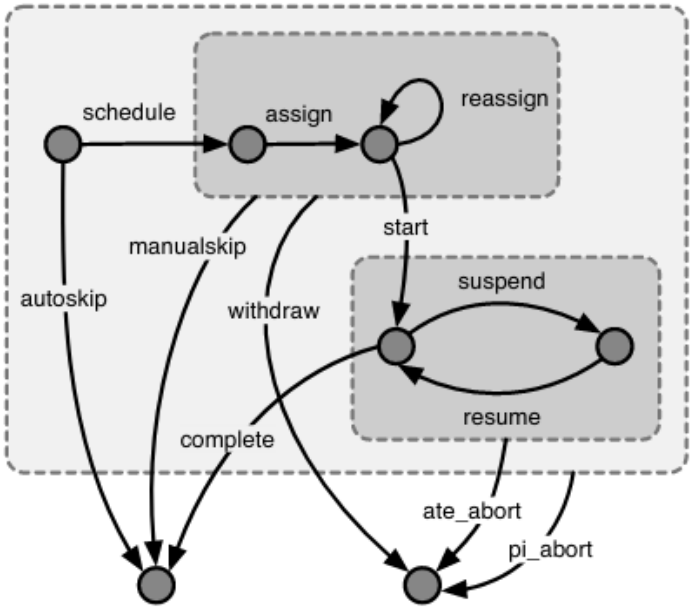

Figure 2.3: Standard XES Transactional Model

To calculate actual time worked, it is imperative to look at the structure in which life-cycle transitions behave inside a case. One way of visualizing this is by using the flexible heuristics miner, present in ProM 6. In figure 2.2, event lifecycle:transition and event EventOrigin are used as classifiers. It matches the structure of XES Transactional Model (Gunther & Verbeek, 2014), which shows in more clarity the important user steps.

Following from the event logs, it can be determined that there are four steps that are not performed by the system. These steps consist of Start, Suspend, Resume, and Complete. The combinations which contain the actual time worked are:

- Start → Suspend
- Start → Complete
- Resume → Suspend
- Resume → Complete

When one of these combinations is encountered, an employee has spent time on the case, which can be traced back by subtracting the Start/Resume timestamp from the Suspend/Complete timestamp. This is stored in an additional column, which accumulates the actual time worked by employees on that case. This leads to two methods of actual time worked calculation.

The theoretical approach is task-based, which means it only looks at the logged time spent by employees. Therefore it is vulnerable to leaving out important information. This is a naive approach, as we assume the employees log their starting time as they start working on the task. However, the event logs indicate that many tasks are finished within a small time frame of the starting time. This can indicate that employees work on tasks before they log their starting time. If this is the case, not all time the employees work on tasks is recorded. Subsequently, the accumulative time worked on tasks is insufficiently low. To fully encapsulate the actual time worked on tasks, the work schedule of employees has to be taken into account.

The accumulation approach of calculating the actual time worked looks at the time employees spent at the office. The event logs indicate that there are two different work schedules that the users adhere to. Say an employee works from 09:00 to 17:00. This time frame can be used to allocate time to work on tasks. When a life-cycle transition of Complete or Suspend is encountered, the employee has finished working on the task. To allocate this time to the task, a trace-back is necessary to either the ending time of the previous task or the start of the working day. This ensures we encapsulate all time the employee spends at the office. However, it risks overshooting the total time spent on tasks, as it does not take into account the lunch and coffee breaks an employee takes during the day.

The dataset is not free of data quality problems. In the dataset, there are frequent periods of time where a user is not assigned to a certain task. This means when the instanced method of assigning actual time worked is used, the actual time worked is lower than it is in reality. There is no surefire way to assure data quality since it is historical data and it is not possible to verify this afterward.

Another data quality problem is the lack of distinction between users and system events. The dataset consists of users labeled by numbers, but there is a large difference between the speed of operation and the type of operations a user performs. In particular, user_1 performs tasks at inhumane speeds, works all day, and performs mostly system events. It is reasonable to deduce that this user must be a computer, which would have an impact on the actual time worked analysis.

2.3 Data Preparation

This section consists of all the steps taken to reach the final dataset. The data quality problems in section 2.2 are addressed when possible and the dataset is trimmed until only the relevant data is present.
The first step is the removal of User_1. According to the data understanding, User_1 is a computer performing system tasks, thus it is not relevant in our calculation of actual time worked. Using a function to filter data, all traces containing User_1 are removed from the dataset.

Listing 1 Tagging events in between coupled workflow events

```
#Create variables to store name of workflow event and between marker
between = False
st_event_name = 'empty'
    for index, row in df.iterrows():
        if between==True:
            if df.loc[index,'event concept:name'] != st_event_name:
                df.loc[index,'Removable'] = True
            else:
                df.loc[index,'Removable'] = False
                between=False
                st_event_name='empty'
        elif df.loc[index, 'event EventOrigin']=='Workflow':
            between = True
            st_event_name=bpi.loc[index,'event concept:name']
            df.loc[index,'Removable'] = False
        else:
            df.loc[index, 'Removable']=False/
```

A loop is created to mark all events that are contained between a starting and end point, using the between variable. The loop checks if the event is a workflow item and stores the event name when one is encountered. The between variable is also set to True, which leads to a different part of the loop. Here the next events are compared to the stored event name and marked for removal when they differ.

The second step is the coupling of each starting point of a workflow item and its corresponding end point. A loop (Listing 1) is created to mark all events that are contained between a starting and end point, using the between variable. The loop checks if the event is a workflow item and stores the event name when one is encountered. The between variable is also set to True, which leads to a different part of the loop. Here the next events are compared to the stored event name and marked for removal when they differ. This continues until the stored event name is found, between is set to False and the stored name is removed. After this loop runs through the dataframe, every marked event is removed. This ensures every workflow event that starts or resumes, is directly followed by the corresponding suspension or completion.

The last step is the removal of lifecycle transitions: schedule, withdraw, and ate_abort. The XES Transactional Model states these steps are not performed by users, therefore removal of these rows will not impact the actual time worked calculation.

2.4 Modeling

During the modeling stage, the procedure for the creation of actual time worked is discussed. The calculation is done in Python using mainly the Pandas library. The data frame is grouped by either 'case concept:name' or 'event org:resource', when looking at cases or users respectively. The timestamps of each event are shifted, so we create a time_previous_event. When no previous timestamp is able to be shifted, the time of the current event is inserted instead.
Next, the timedelta between the previous and current event is calculated. These timedeltas are filtered on negative or extremely large values, which indicate that either the previous event is not part of the same trace or more than a day has passed. These timedelta would lead to a false conclusion of the actual time worked. Subsequently, for the {first} method, all timedeltas of events that are not 'event lifecycle:transition' 'suspend' or 'complete', are set to zero. This leads to the filtering of all events that users did not spend time on. Lastly, the cumulative sum of the timedeltas is taken in both the theoretical and accumulation methods. The total_seconds function is used to create a numeric value. A division of 3600 is done to increase readability, as the actual time worked is transformed from seconds into hours.
To create the throughput time, the starting time of a case has to be inserted into the dataframe. This is done by setting the 'start_time' to the 'event time:timestamp' and nulling out all non-starting indexes of the cases. This null is then filled using the previous accepted datapoint. The throughput time is then calculated by subtracting the 'start_time' from the 'event time:timestamp'. This timedelta is transformed into a numeric value using the total_seconds function and a division of 3600 is used to transform this into hours.

2.5 Evaluation

This stage contains the validation of the created model.The goals are to check if the model is of high quality and if the objectives of the model are reached. The ground truth is the true value of actual time worked. The dataset used consists of real life data. This means there is no ground truth present, thus there is no way to validate both methods of actual time worked calculation. A own dataset is created to achieve this goal. The ground truth is inserted in this created dataset, making it possible to check the validity of both calculation methods. The dataset consists of two days of a average user, where the ground truth is inserted as time worked for every entry in the log. This includes time spent on breaks and time not spent working. This gives a baseline for both methods to be evaluated from, which would show the differences and validity. Another method of evaluation is achieved through data inspection and visualizations. The final dataframe created with the methods can be inspected to check the accumulated times and timedeltas for anomalies.

2.6 Deployment

This stage is not discussed in the paper, as this method is not tested or validated in a real-life situation. The deployment stage usually contains an organization of knowledge and is presented in such a way that the customer can use it. This can be done by the presentation of a report of a repeatable data mining process. In the event of a deployment taking place, the presentation of this research would take the form of both a report and a repeatable data mining process. The feedback of this stage is used to further improve business and data understanding, which in turn improves the model.

3 Results

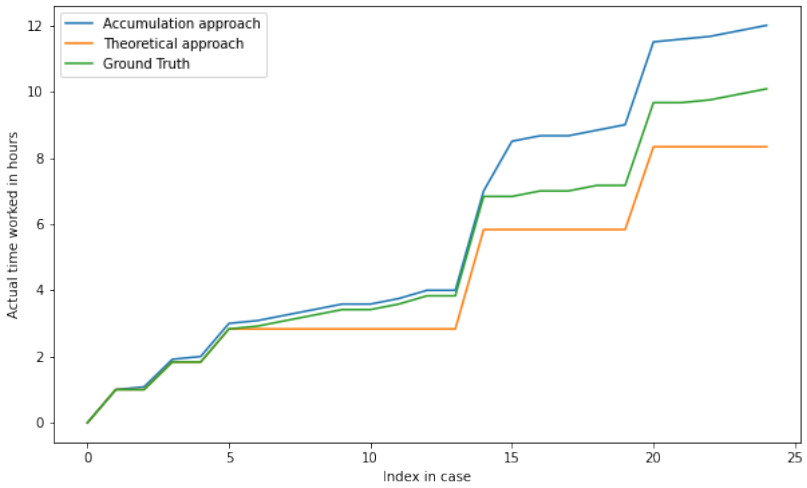

Figure 3.1: Ground truth comparison

In figure 3.1, the created dataset containing the ground truth is used. The X-axis shows the index in the case, referring to the events in the case. This plot shows the result of 25 events. The Y-axis shows the actual time worked for this user in hours. A comparison is made between the two methods and the ground truth, giving insight into the calculations and making it possible to evaluate both methods. The accumulation approach shows the highest amount of time worked, the theoretical approach shows the lowest amount, and the ground truth falls between these two approaches. The figure shows that the accumulation approach forms a upper bound for the ground truth, where the theoretical approach forms a lower bound.

To answer sub-question 1.1: The difference between the theoretical and accumulation approach is rooted in the approach to tackling parallel task processing. The accumulation approach allocates all time where the user was present in the system to this user. This includes lunch breaks, time not spent working, and ineffective time. This naturally leads to an overestimation of the actual time worked. The theoretical approach only includes time that is worked inside a logged workflow. It starts logging when a workflow is started or resumed, and stops when this is suspended or completed. The resulting time is highly accurate, but users also perform work outside these workflows and users log their data imprecisely. It naturally follows that this leads to underestimation of the actual time worked. As is shown by figure 3.1, the ground truth falls between the two bounds of the accumulation and theoretical approach.

To answer sub-question 1.2: How can the actual time worked calculation be verified? The dataset used does not contain a ground truth, meaning there is no way to evaluate and verify the calculations of actual time worked. The solution for this is to create a dataset where the ground truth is inserted, as explained in section 2.5. This gives a baseline to verify the methods and gives insight into the differences of both approaches.

The actual time worked gives insight in the type of users are present in the company. In figure 3.2, four users are taken as example of the four types. The actual time worked (theoretical approach) is shown for four users. Each user shows a different pattern of behaviour. The users differ on two aspects, one being the consistency in weeks which they work. Some users work every single week, while some have large breaks in between weeks. The second aspect is the amount of hours worked in each week. Some users work a high amount of hours in the week, while other users work a low amount of hours in the week. The difference in behaviour is likely due to the type of work users perform in the system. The profile

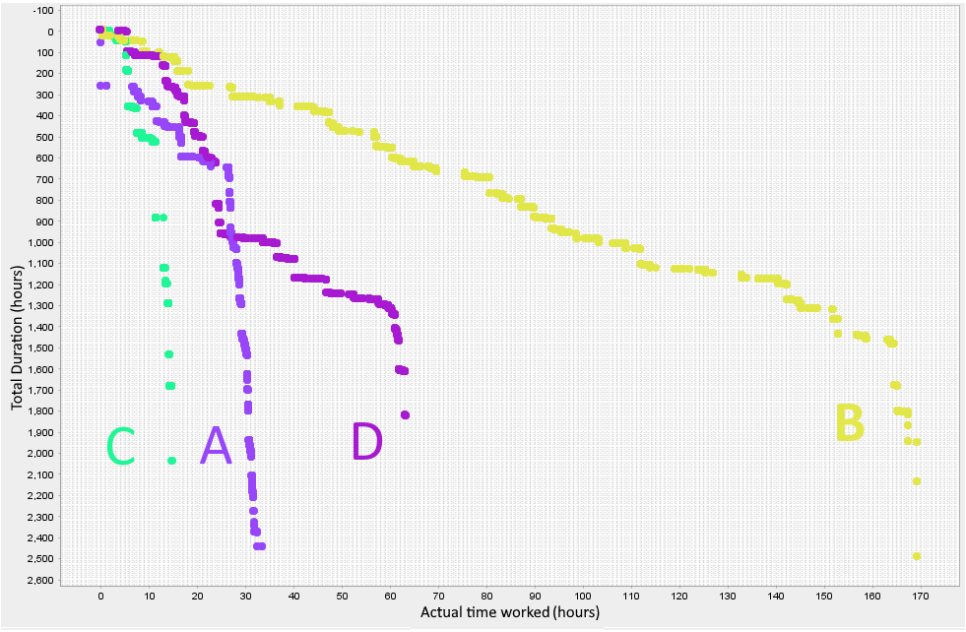

Figure 3.2: Four Users profiles

types provide a general overview of the behaviour of users. These four types are additionally visualized in figure 3.3. A consistent user with a low amount of hours worked per week is shown in figure 3.2, which is the dark purple line marked with type A. A consistent user with a high amount of hours worked is marked as type B. A sporadic user with a low amount of hours worked is marked with type C, and a sporadic user with a high amount of hours worked is marked by type D, all of which are present in the figure. The type of user profiles obtained from the data can be of great assistance in improving the effectiveness of employees. Therefore, it is interesting to investigate additional work in the automation of this profiling process for all employees.

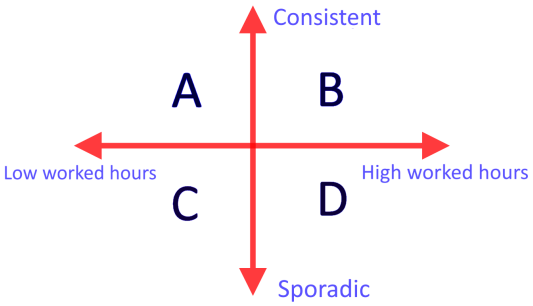

Figure 3.3: Profiles of users

To answer the main research question: How can the actual time worked by employees be determined from Event logs? The proposed methods, the accumulation and theoretical approach, lead to a upper and a lower bound of the ground truth respectively, as shown in Figure 3.1. The actual time worked must lie in between these two calculations. The actual time worked is not present in the dataset, but these two bounds represent a step closer to the actual time worked. Additional work must be performed to provide closer bounds and a consistent estimation of the actual time worked.

4 Discussion

The results from Figure 3.1 show that the two approaches lead to a upper and lower bound of the actual time worked. A first intuition is that a combination of these two bounds likely leads to a consistent estimation of actual time worked. Additional work must be done to verify this claim. The profile type of users also has an effect on the tightness of the bounds. Users that work consistent and have high amount of worked hours, thus falling in type B, provide tighter bounds than users that work consistent with low amount of worked hours. There is simply more information to attribute worked time, leading to a tighter lower bound. There is also less time in between tasks, leading to a tighter upper bound.

The profile types provide more insight into the tightness of the bounds. Further work can be done on automatic profile detection. The consistency of users can be quantified, just as the amount of worked hours for every week. Users can be automatically classified into the four categories.

4.1 Threats to validity

The validity of the actual time worked calculation is based on reasoning and anomaly checking. This gives a strong enough basis to derive a meaningful statistic, but it does not provide sufficient proof that it always holds. In this section, further threats to the validity of actual time worked calculation are discussed.
The first threat of validity concerns the data input of the company. To track the time spent on certain tasks, it is crucial to ensure correct and punctual time entry of the start and end of tasks. On inspection of the data set, there seem to be significant gaps between the ending of a task and starting the next one. This indicates that users either take long breaks between registration and closing of a task, or the input of the data is incorrect. When looking at individual cases, it seems that the data input is incorrect. Since there is no reliable way to check whether this is the case, one either has to ensure correct data input or have the process mining and data science skills to judge the data quality.
The second threat to the validity comes in the form of parallel task processing. The most straightforward way of measuring actual time worked is when users work on one case until it is done. This leads to a one-on-one relationship between cases and users. In the data set used in this research, this is not the case. Users can switch between cases and perform one in many tasks in a case. Switching between cases should not give problems in analysis, but when introducing questionable data input, one has to make choices on how to attribute the time spent.
The final threat to the validity lies in the nature of the data set. Validity is dependent on the data set and by extension the company that collects the data. Data sets consist of different cases, tasks, and events present, which lead to a different approach when calculating actual time worked. The attribution of time spent working differs when dealing with different types of tasks. To combat this, one needs to have either process mining or data mining expertise present to make differences visible and present a strategy to tackle these differences.

4.2 Ethical Concerns

When dealing with data related to people, it raises ethical concerns about the usage of said data. The created statistic actual time worked can be used to verify the working times and completed work of employees. This can lead to negative effects on employees, such as feelings of privacy invasion, perceptions of unfairness, decreased job satisfaction, and lower task performance and productivity of less-skilled workers (Tomczak et al., 2018). The 2018 paper presents strategies to maximize the positive effects and minimize the negative effects of performance management. Employers have to strive to "be transparent about the usage of performance management" and be "aware of all potential employee reactions to being monitored".

4.3 Future Work

One large improvement to this research is the real-life testing of the actual time worked statistic. This can be achieved by asking employees of a company to track the time they spent on tasks during the day. This additional information is used to check for conformance with the actual time calculation. Additionally, the proposed methods have been tested on one data set only. Thus to achieve a broader overview and better grasp of its validity, the calculation has to be applied to multiple data sets, ideally from different fields and using different task structures.
Since this implementation is quite bare-bones, the path to a usable product is still very long. This research looked into the actual time worked, but the use case within a company requires another approach. Actual time worked would have to be instanced to increase visibility and automatic outlier calculation would add more indicators for employers to effectively counsel their employees.

5 Conclusion

This paper presents two methods for the calculation of actual time worked. Actual time worked is a statistic that takes the context of parallel task processing into account and provides more insight into the behaviour of users and case progression. The Crisp-DM framework is used to develop this statistic. The different approach in tackling parallel task processing leads to two methods, the accumulation method and the theoretical method. The main difference between the two methods is the allocation of break times and time not spent working. The verification of these methods are achieved by inserting the ground truth in a created dataset. Results show the accumulation and theoretical methods provide an upper and lower bound respectively for the calculation of the actual time worked. This answers the main research question of this paper: How can the actual time worked by employees be determined from event logs? Additionally, this actual time worked calculation provides insight in the profile type of users. Future work can be done on combining the two methods to provide more consistent estimation of actual time worked. There is also the possibility of automation of profile types.

References

Gunther, C., & Verbeek, H. (2014). *Xes - standard definition*. BPMcenter. org.

Hall, B. L., & Hursch, D. E. (1982). An evaluation of the effects of a time management training program on work efficiency. *Journal of Organizational Behavior Management*, *3*(4), 73-96. Retrieved from https://doi.org/10.1300/J075v03n04_08 doi: 10.1300/J075v03n04_08

McConnell, C. R. (2004). Managing Employee Performance. *The Health Care Manager*, *23*(3), 273–283. doi: 10.1097/00126450-200407000-00012

Moreira, C., Haven, E., Sozzo, S., & Wichert, A. (2018, 12). Process mining with real world financial loan applications: Improving inference on incomplete event logs. *PLOS ONE*, *13*, e0207806. doi: 10.1371/journal.pone.0207806

Tomczak, D. L., Lanzo, L. A., & Aguinis, H. (2018). Evidence-based recommendations for employee performance monitoring. *Business Horizons*, *61*(2), 251-259. Retrieved from https://www.sciencedirect.com/science/article/pii/S0007681317301611 doi: https://doi.org/10.1016/j.bushor.2017.11.006

Weijters, A., & Ribeiro, J. (2011). Flexible heuristics miner (fhm). *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 310-317.

Wirth, R., & Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining. , *1*, 29–40.

Appendix

Code for actual time worked calculation

```
import pandas

#Function to import csv file into pandas dataframe
def import_csv(file_path):
    event_log = pandas.read_csv(file_path, sep=',',
    parse_dates=['event time:timestamp'],
    infer_datetime_format=True, dayfirst=True)
    return event_log

#Set floating point format and import the data into dataframe
pandas.set_option('display.float_format', lambda x: '%.5f' % x)
bpi = import_csv("data.csv" )

#filter values from dataframe
def filter_data(df,column, value):
    dfb = df.drop(df[df[column]==value].index)
    return dfb
#Filter unuseful data
bpi = filter_data(bpi,"event org:resource", "User_1")
bpi = filter_data(bpi,"event lifecycle:transition","schedule")
bpi = filter_data(bpi,"event lifecycle:transition","withdraw")
bpi = filter_data(bpi,"event lifecycle:transition","ate_abort")

#Define names for variables
user = "event org:resource"
delta = 'time_worked'
timestamp = 'event time:timestamp'

def calculate_user_times(df):
    #Create variables to store name of workflow event and between marker
    between = False
    st_event_name = 'empty'
    for index, row in df.iterrows():
        if between==True:
            if df.loc[index,'event concept:name'] != st_event_name:
                df.loc[index,'Removable'] = True
            else:
                df.loc[index,'Removable'] = False
                between=False
                st_event_name='empty'
        elif df.loc[index, 'event EventOrigin']=='Workflow':
            between = True
            st_event_name=bpi.loc[index,'event concept:name']
            df.loc[index,'Removable'] = False
        else:
            df.loc[index, 'Removable']=False
```

```
    df.reset_index(drop=True, inplace=True)
    #insert time previous event into every event and calculate timedelta
    df['time_previous_event'] = df[timestamp].shift().fillna(df[timestamp])
    df['time_previous_event'] = df['time_previous_event']
    df[delta] = df[timestamp]-df['time_previous_event']

    #insert previous lifecycle transition into every event
    df['previous_event'] = df['event lifecycle:transition'].shift()

    #Filter faulty timedelta values
    df[delta] = df[delta].apply(lambda cell: pandas.Timedelta(seconds=0) if
    (cell.total_seconds()<0 or cell.total_seconds()>(36000)) else cell)

    #Calculate cumulative sum of actual time worked unfiltered
    df["accumulated_user_times"] = df[delta].cumsum()
    df["accumulated_user_times"] = df["accumulated_user_times"].apply(
    lambda cell: cell.total_seconds()/3600)

    #Filter based on XES transactional model
    df.loc[(df['event lifecycle:transition'] != 'suspend') &
    (df['event lifecycle:transition'] != 'complete')
    , delta] = pandas.Timedelta(seconds=0)

    #Set Application and Offer events to zero
    df.loc[df['event EventOrigin'] == 'Application',
    delta] = pandas.Timedelta(seconds=0)
    df.loc[df['event EventOrigin'] == 'Offer',
    delta] = pandas.Timedelta(seconds=0)

    #Calculate cumulative sum of actual time worked filtered
    df["acc_user_times_filtered"] = df[delta].cumsum()
    df["acc_user_times_filtered"] = df["acc_user_times_filtered"].apply(
    lambda cell:cell.total_seconds()/3600)

    #Insert case index and starting time for every task
    df['case_index'] = df.index
    df['start_time']= df['event time:timestamp']
    df.loc[(df['case_index']!=0), 'start_time'] = None
    df['start_time'].fillna(method='pad', inplace=True)

    #Calculate throughput time using start_time and current timestamp
    df['throughput_time'] = (df[timestamp]-df['start_time'])
    df['throughput_time'] = df['throughput_time'].apply(
    lambda cell: cell.total_seconds()/3600)

    return df

#apply function calculate user times to dataframe
bpi_work_task = bpi.groupby('case concept:name').apply(calculate_user_times)
bpi_work_user = bpi.groupby(user).apply(calculate_user_times)
```

```
#Export modified dataframe to csv file
bpi_work.to_csv("data_work_task.csv", sep=";", index_label="index")
bpi_work.to_csv("user.csv", sep=";", index_label="index")
```