Start Time and Duration Distribution Estimation in Semi-Structured Processes

Andreas Wombacher and Maria-Eugenia Iacob

University of Twente, The Netherlands
a.wombacher@utwente.nl, m.e.iacob@utwente.nl

Abstract. Semi-structured processes are business workflows, where the execution of the workflow is not completely controlled by a workflow engine, i.e., an implementation of a formal workflow model. Examples are workflows where actors potentially have interaction with customers reporting the result of the interaction in a process aware information system. Building a performance model for resource management in these processes is difficult since the required information is only partially recorded. In this paper we propose a systematic approach for the creation of an event log that is suitable for available process mining tools. This event log is created by an incrementally cleansing of data. The proposed approach is evaluated in an experiment.

1 Introduction

Semi-structured processes are business workflows, the execution of which is not entirely controlled by a workflow engine, i.e., an implementation of a formal workflow model. Example of such processes are workflows in which people interact with clients and/or paper documents in order to insert, approve, or validate information in an (web-based) information system. Such an information systems can be an application server or a service orchestration, e.g., using BPEL. To support a better understanding of the management of such processes it is important to assess the performance of activities in the workflow and their relations to available resources. Lacking such knowledge makes it hard to predict the utilization of resources and to make a balanced resource planning. For example, it is difficult to predict how well the business can cope with higher workload due to, for example, activity peak, a promotion activity or to vacations.

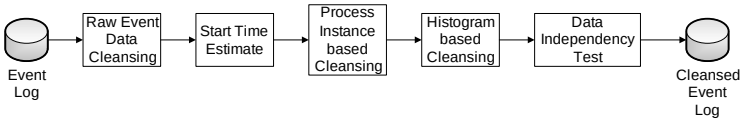

Fig. 1. Cleansing rules overview

Independent of the workflow's implementation, the underlying information system may keep track of the completion time of an activity but cannot record the start time of an activity. Such an information system cannot detect for instance when a conversation with a client starts. Thus, it is not possible to build a classical performance model and use existing process analysis techniques like those described in [1] before enriching the data with the activities' start times. Therefore, in [2] we proposed an approach for cleansing the data and estimating the activity start times based on the steps depicted in Fig 1. A first cleansing step is performed on the raw event data eliminating data which are unusable due to infrastructure problems (e.g., network problems). Next, the cleansed data is used to infer an initial estimate of the start time for each activity. The initial estimates may be overwritten in later cleansing steps. The following cleansing step investigates special situations per process instance (also called case) like, for example, system tests and dead-lock or live-lock instances. The last cleansing step is the histogram based cleansing (per activity) that removes outliers, i.e., exceptionally high durations of an activity. The final step investigates dependencies of activity durations cross process instances and categorical data (e.g., the weekday or the experience of a user), thus checking whether the independence assumption used in a performance model is actually supported by available data. The final result is a cleansed event log, which can be used for the mining of a control flow and for performance analysis.

In [2] we have reached the conclusion that the inferred performance estimates for the case study were quite low compared to the expected performance measures. Therefore, in this paper we improve and extend the results of [2]. Thus, the contribution of this paper is twofold. First, we present additional insights into start time estimates including a mathematical description of the problem. Second, a new approach for histogram based cleansing is proposed. Furthermore, the findings will be evaluated on synthetic datasets. To the best of our knowledge, our approach is the first to attempt the estimation of the duration distribution of activities using logs that contain incomplete information regarding the process execution.

The paper is organized as follows. We first present the context of the problem and its description (Section 2). The start time estimation approach and histogram based cleansing are presented in Section 3 and Section 4 respectively, followed by an evaluation (Section 5). We conclude the paper with related work (Section 6) and conclusions (Section 7).

2 Problem Description

The proposed approach has been motivated by the semi-structured processes in the front-office of a financial company. The service provider uses a web-based application to quickly set up semi-structured financial processes without developing the same components repetitively. A typical front office employee handles applications of clients for, e.g., a loan, insurance or savings account, at the office counter, but also Internet and telephone applications. Typical activities in the

front office are talking to the client, collecting and verifying client documents, do some automatic checks (e.g., a credit check), and sending the application to the back office for further handling. The development environment uses a proprietary process modeling language based on states, and manual and automatic state changes, performed by an employee or by the software. The expressiveness of the language is comparable to that of Finite State Automata, i.e., it supports loops but no parallelism. Thus, the system can only record an activity's completion time (state change), but not its start time.

The challenge posed by such semi-structured processes is that start times of activities cannot be automatically logged. Another issue is that users often work on more than one process and therefore the percentage of time a user is working on the process under investigation is unknown. Furthermore, other activities like, e.g., meetings, breaks, early departure of an employee are not documented and therefore are not available for the start time estimation.

In this paper we assume the existence of a process execution log file, which contains information about the case ID, the State Change ID, the Completion Time, the ID of the user performing the state change, and the name of the activity being completed. The State Change ID provides a complete order on all state changes. The Completion Time provides a partial order of state changes. An example of a log file is depicted in Fig 2, which will be also used later in the paper, and is partly visualized in Fig 3.

Case ID	State Change ID	Complet. Time	User ID	Completed Activity
2	3	9:44:14	Andy	Process Start
1	4	9:49:54	Andy	Send Request
1	5	10:15:00	Peter	Control Opening
1	6	09:05:00	Andy	Credibility Check

Fig. 2. Example State Transition Log

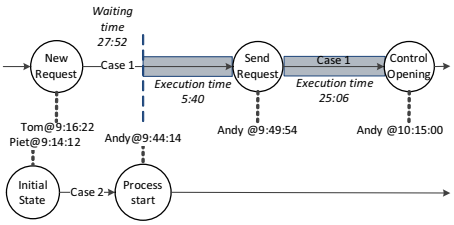

Fig. 3. Start time inference

In the following we assume that the process involves, potentially, multiple systems each providing part of the log information. However, we are not addressing neither data integration problems (e.g., entity resolution problems of event log information) nor syntactic or semantic data integration problems. In particular, in the following sections, we are discussing start time estimation and histogram based cleansing.

3 Start Time and Duration Distribution Estimation

In this section we propose an approach for estimating the start time of activities, which is required to determine the duration of an activity. In this section, we extend the basic idea presented in [2] by explaining the chunks of time a user has worked on an activity. Further, potential issues are discussed and a mathematical model capturing these issues is proposed.

3.1 Approach

Estimating the start time of an activity is based on a complete order of state changes (activities), which is consistent with the partial order of the Completion Time. First, the control flow dependencies in a workflow ensure that an activity can only start after the preceding activity has been completed. Thus, by determining the Completion Time of the preceding activity an estimate of the start time of the activity can be inferred. With regard to the example in Fig 2 the activity *Control Opening* has the preceding activity *Send Request*. Thus, an estimate for the start time of the *Control Opening* activity is the completion time of the *Send Request* activity. This results in an estimated processing time or duration of 25 minutes and 6 seconds as depicted in Fig 3.

Second, we make the assumption that a user can only perform one activity at the time. Thus, an activity performed by a user can only start after another activity performed by the same user has been completed. With regard to the example in Fig 2 the activity *Send Request* of case 1 performed by user Andy is preceded by the completion of activity *Process Start* of case 2. Thus, an estimate for the start time of the *Send Request* activity is the completion time of the *Process Start* activity. This results in an estimated processing time or duration of 5 minutes and 40 seconds as depicted in Fig 3.

Thus, the estimated start time of an activity is the minimum of the completion time of the preceding activity of the same process, and the completion time of the preceding activity of the same user. Consequently, the start time of the first activity in a process can only be estimated based on the preceding activity of the same user since there is no preceding activity in the process. Applying this definition, as also given in [2], results in significantly underestimating the processing times of activities. This is due to the possible interweaving of the execution of activities performed by the same user. Therefore, the approach is adjusted as follows.

The initial start time estimate is the completion time of the preceding activity of the same process. If no such activity exists the completion time of the preceding activity of the same user is chosen. From this initial start time estimate the estimated processing times of preceding activities performed by the same user in the time span of the initial start estimate and the known completion time are subtracted. This provides a more accurate processing time/duration estimate.

The proposed approach is illustrated in Fig 4. The start time of activity 2 (Act 2) is initially estimated by the completion time of the preceding activity of the same process, which is activity 1. Thus the duration of the activity is the

difference between completion and start time. From this initial duration estimate the duration of chunks are removed, which fall into this processing time interval and which have been associated to executing other activities by the same user. A chunk is represented by a gray box in the figure. In this case, the processing of activity 2 is performed in two chunks: the first one is the non assigned time of user 2 between activity 5 and 3 and the second one is the non assigned time between activity 4 and 2. The more processes are interwoven the higher the number of chunks.

3.2 Potential Issues

In this section several issues are raised that can not be resolved by the start time estimate. They are dealt with by histogram based cleansing (see Section 4) by making sure that they do not influence the performance estimates significantly.

Working Hours of Users. A challenge for start time estimation of activities is that working hours are not precisely fixed. Let's say Jim completed the last activity on Tuesday at 17:00 and the next activity completion is Wednesday at 9:05. This doesn't mean that Jim took 16 hours and five minutes to complete a task. Since there is no information about the start and end time of an employee's working day, there is no possibility to provide a better start time estimate. However, it is reasonable to assume that activity durations influenced by this issue are longer than at least 8 hours.

Non-visible Activities. In the proposed approach we implicitly assume that a user is only working on the system under investigation. However, a person also performs other tasks in addition to working in this particular system. For example, when user Jim completes the state change 'send request' at 09:48, then attends a meeting till 11:00, and then completes the state change 'control opening' at 11:05, the system will assume that it took Jim 65 minutes to execute state change 'control opening', instead of the actual five minutes work. We call such activities non visible activities, since they are activities of the user, but they are not documented in the event log. Since there is no information available on how much of its time a user is working on the system under investigation, it is not possible to improve the start time estimates. However, it is reasonable to assume that the number of activities performed in the system under investigation is sufficiently high to be able to perform statistical investigations. Furthermore, it is assumed that there is a sufficient number of activities which are not influenced by non-visible activities, otherwise the non-visible activities become so dominant that the performance model describes the non-visible activities rather than the activity under investigation.

Preemption of Activities. The example used in Section 2 for determining the start time works fine if the activities are not preempted, i.e., interrupted to perform another activity. Examples of preemption are a call from your supervisor to drop everything and perform another activity, as depicted in Fig 5. The activity 'Send Request' of case 1 is interrupted by activity 'Process start' of case 2. As a consequence, the estimated processing time of activity 'Send Request' of case 1 is 5:40 (see Fig 3) rather than the actual 7:30. It seems safe to assume

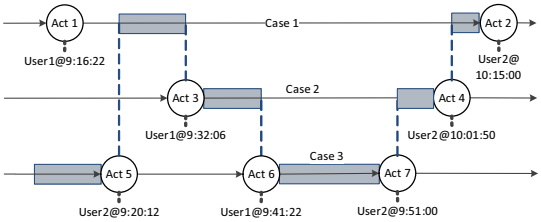

Fig. 4. Adjusted start time inference

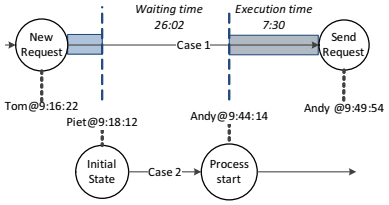

Fig. 5. Preemption of activities

that the probability of preemption is rather low and that the point in time of the preemption to happen is equally distributed.

3.3 Mathematical Model

The basic idea is to describe the duration of an activity as a mixture model combining several distributions with different characteristics and different probabilities of occurrence into a single distribution model. In general, if the number and type of used models and their characteristics is known it is possible to infer the characteristics of these models. However, this is not the case in the problem addressed in this paper. Nevertheless, if the duration of the activity is the dominant distribution and the characteristics of the distributions describing issues addressed above are sufficiently different, it is possible to infer the characteristics of the dominant distribution with an acceptable error.

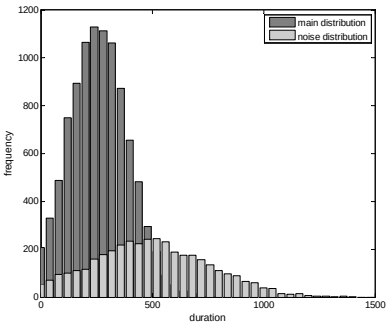

Fig. 6. Histogram of activity execution durations and a noise distribution

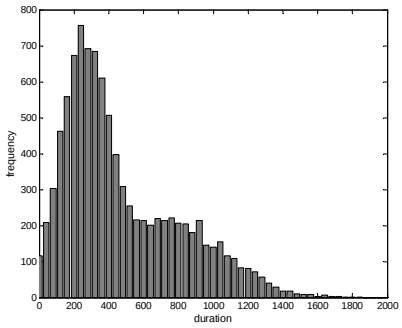

Fig. 7. Histogram of activity execution durations combined with a 40% chance of noise

As an example, Fig 6 depicts two normal distributions where the duration of the activity is modeled as the main distribution and an issue is modeled as a noise distribution. The main distribution has a mean of 275 and the noise distribution

a mean of 500. Both are normal distributions with a standard deviation of 275. The probability of noise is 40%. Fig 7 depicts the resulting mixed model, where the peak is around 300 and a small second peak is observable at around 750.

Since no assumptions are made on the type and number of distributions used in the mixed model, the aim is to determine the mean of the most dominant distribution. With regard to the example, the aim is to determine the peak in Fig 7. In general it can be stated that the dominant mean estimation is most precise if the influence of the noise distribution is minimal. This is guaranteed in case the noise probability is low or the noise distribution has a significantly different mean value. An example for a low noise probability is that the number of coffee breaks a clerk has per day is at least a magnitude lower than the number of performed activities in the system. An example for a significant different mean value is the working hours of users, i.e., the durations of activities started on one day and completed on the following day have a noise of at least 8 hours which is at least a magnitude higher than the actual duration of the activity.

For a given main distribution $\Theta(\mu, \dots)$ with a mean μ and some other parameters, and a set of error distributions $\theta_i(\mu_i, \dots)$, $i = 1, ..., n$, the mixed distribution Θ' can be defined as

$$\Theta'(\mu') := \Theta(\mu, \dots) + \sum_{i=1}^{n} p_i * \theta_i(\mu_i, \dots),$$

where $0 < p_i \leq 1$ is the probability of an error distribution θ_i. Since no further assumptions are made on the distributions and of the error probabilities, the formula stays this general. Further, the mixed distribution is only characterized by the mean μ', since this is the parameter inferred in the following section.

4 Histogram based Cleansing

In this step of the process presented in Fig 1 the histograms of activity durations with the same label over all process instances are investigated. The duration is defined as the difference between the Completion Time of an activity and its estimated start time. The basic idea is that the distribution of durations per activity is a mixed model (see Section 3.3). The aim is to find the mean of the dominant distribution. Therefore, the observed durations of an activity - ordered by durations - have to be clustered. In particular, the duration of the centroid of the biggest cluster corresponds to the mean of the dominant distribution. In the remainder of this section three potential methods are presented and compared.

k-Means Clustering. A well known clustering approach is k-means clustering [3], which consists of the following steps: 1) estimate the number of clusters which should be considered, 2) perform a k-means clustering, and 3) determine the largest cluster and use the cluster centroid as a mean of the dominant distribution. The first challenge is to estimate the number of clusters to be used. Different approaches are proposed in literature, such as, the elbow method, where the smallest number of clusters is chosen, which does not improve a cluster quality measure. Other approaches are either based on internal or external measures

of the clusters. In the following (and especially in the evaluation) the optimal number of clusters is derived from the construction of the evaluation data set. The mean of the main distribution is determined by the centroid of the largest cluster determined by k-means. The largest cluster is chosen since the assumption is that the main distribution is dominant and therefore the number of durations around the mean of the main distribution is much bigger than for the means of the error distributions in the mixed model.

Kernel Density Estimation. The basic idea is to build a histogram of all observed durations of an activity and to use the bin in the histogram with the highest frequency as the mean of the main distribution. Since such an approach is dependent on the definitions of the bins in the histogram, an approach for estimating the probability density function is used, which is called kernel density estimation [4, 5]. The inferred kernel can then be used to create the density value for the complete domain. The mean of the dominant distribution is the maximum in the probability density function. Please note that this approach does not use the estimates of the mixed model but the resulting estimated distribution created by the mixed model.

Slope based Clustering. The basic idea is that the dominant distribution has the strongest representation in the observed durations. Therefore, the aim is to find the largest subset of observed durations where the slope or gradient between two subsequent durations does not exceed a specific value. Further, the subset must contain at least 5% of all observed durations.

Formally, this can be described as follows: The observed durations can be described as an ordered set $X = \{x_0, \ldots, x_N\}$. The mean of the main distribution corresponds to the mean of the largest subset $X' \subseteq X$ with $X' := \{x'_0, \ldots, x'_n \in X | \forall j \in \{1, \ldots, n\}, x'_j - x'_{j-1} < \varepsilon\}$ of observed durations with the lowest threshold $\varepsilon \in \{0.1, \ldots, \frac{n}{2}\}$, where the size of the subset X' must be significant, thus it must contain more than 5% of the total size of the data set $n > 0.05 * N$.

Discussion. The initial testing and comparison of the different approaches indicate that all of them are sensitive to the actual observed durations. Therefore the evaluation was repeated 20 times and the average errors were considered. Each experiment was based on 10000 observed durations, where the probability of noise has been kept constant (40%), while the noise and main distribution means vary. The evaluation has been performed for a Normal distribution with a fixed standard deviation of 275 and a Poisson distribution. The results are given in Table 1. It turns out that the differences in the test using Poisson distribution are smaller and very different from the test with Normal distribution. For the Poisson distribution the slope based clustering performs best, closely followed by the kernel density estimation. The k-means clustering is in most cases comparable to kernel density estimate and slope based clustering, and in a few cases it is quite off. There is no apparent reason behind the distribution of the higher error cases. In case of a Normal distribution, the average absolute errors are depicted in Fig 8. It shows that the kernel density estimate approach outperforms the other two approaches as it can also be seen in Table 1. The slope based approach has always the worst results, while the k-means clustering

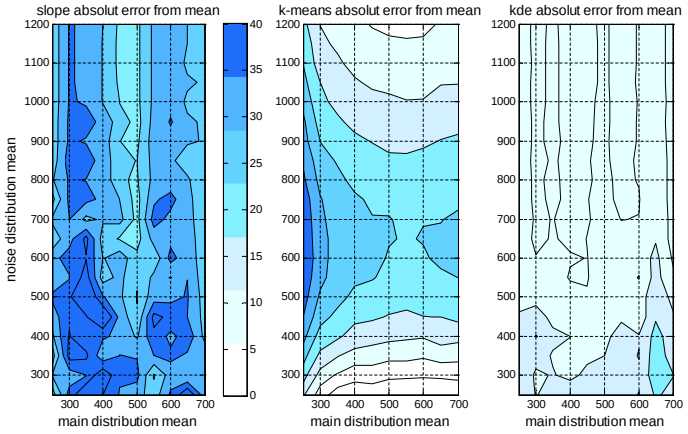

Fig. 8. Contour plots of the absolute errors of the k-means and the clustering method for varying mean values of the main and error distribution

Approach	**Normal Distribution**			**Poisson Distribution**		
	Min	Mean	Max	Min	Mean	Max
slope based clustering	22.0	31.7	46.3	0.1	0.2	0.3
k-means clustering	1.8	19.8	39.4	0.1	1.2	24.2
kernel density estimation	7.5	12.0	22.8	0.4	0.9	3.1

Table 1. Summary of Estimated and Guessed Durations

approach performs well with very low and very high means of the noise distribution. Further evaluation will use the kernel density estimate approach for arbitrary distributions and the slope based approach for Poisson distributions only.

5 Evaluation

In this paper we focus on an evaluation with synthetic data since this allows us to evaluate our approach in various situations. In [2] and [6] we have reported two case studies illustrating that our approach can also be applied in real situations.

The evaluation with synthetic data requires a generic method for generating log information for arbitrary workflows with varying noise conditions. Therefore, the Colored Petri Net Tool (CPNTool[1][7]) is used with an extension to log process execution information in a file [8]. The logged information can then be used to reconstruct the exact execution of the workflow for all cases. By using a subset of the data as input for the presented approach, the estimated

[1] `http://cpntools.org/`

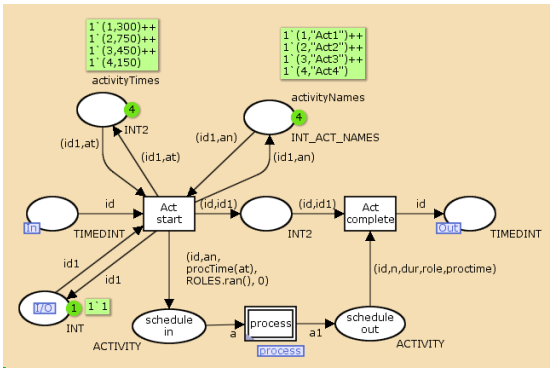

Fig. 9. CPN subnet activity

performance information can be compared with the actually used mixed model, encoded in the workflow specification. Next, we first explain the used hierarchical Petri Net and then we present the evaluation results.

5.1 Workflow Specification as a hierarchical timed Colored Petri Net

A Petri Net [9] is based on transitions (squares) and places (circles) which are connected as a bipartite graph. A Colored Petri Net allows to assign types to places and variables of those types to arcs which are connecting places and transitions. A hierarchical Petri Net allows to define subnets where executing a transition (represented by a square with double border) on the higher level means executing the subnet on the lower level. The input and output places of the higher level transition have to be mapped to the input and output places of the subnet. A timed Petri Net allows to specify a delay for the execution of a transition based on a time model followed during the execution of the net. An delay of 12 time units is represented in the model by annotating an arc with @+12. Examples of these nets are depicted in Fig 9, and 10 representing the modeling of a single activity, and an execution engine processing scheduled activities. The used Petri Net consists out of four hierarchical levels. The generator creates a specified number of instances and provides a randomly distributed initialization of workflow instances. The workflow consists of four sequential activities, which is a simple example, but represents the current state of our research. In future work we will investigate more complex workflows. An *activity* transition is annotated with the activity ID used in the further processing (id1 variable). The id variable contains the workflow instance ID. Arbitrary workflows can be realized. Each *activity* transition executes the activity subnet depicted in Fig 9. The *Act start* transition determines the mean processing time (*at*), the name of the activity (*an*), the actor executing the activity (*Roles.ran()*), and the actual

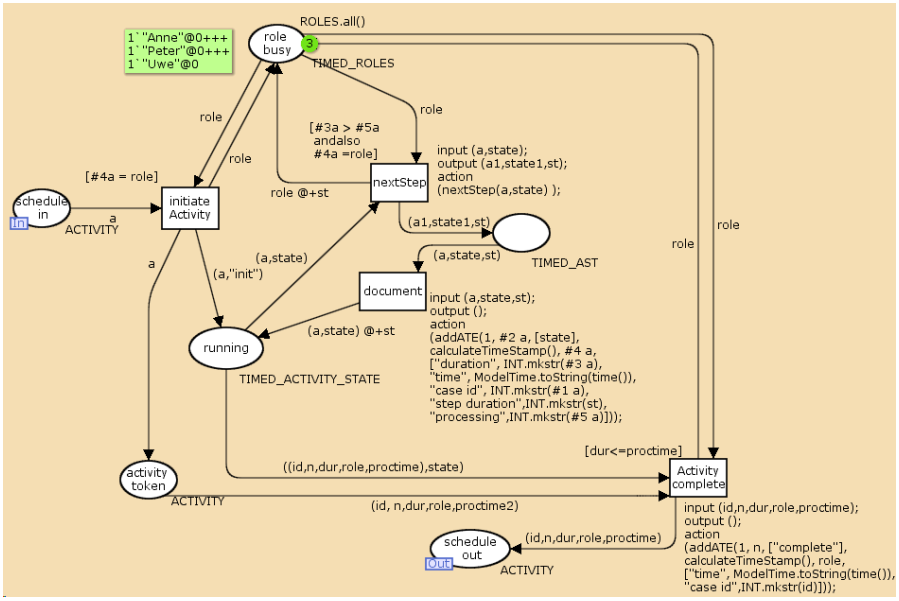

Fig. 10. CPN subnet process

processing time of the activity (*procTime(at)*) and schedules the activity. The *process* activity executes the process subnet.

The process subnet (representing a workflow engine) (see Fig 10) ensures that an actor can only perform an activity at a time. Further, it documents state changes of an activity execution (*document* transition) and the completion of an activity (*Activity complete* transition) using the *addATE* function described in [8]. The *nextStep* transition determines in which state the execution of an activity continues. States are actually executing the activity at hand ('running'), having a break ('break'), attending a meeting ('meeting'), and working outside the system under investigation ('external') with clear defined probabilities. These states have been chosen because all states have different characteristics: breaks are usually shorter than meetings, which are shorter than external activities. In case there are breaks which have a length of a meeting, this means that the probability of having a meeting should be increased. The core is that the reason for the interruption of an activity execution is irrelevant as long as a specific characteristics of a case is preserved. In terms of the mixed model described in Section 3.3, the activity processing is the main distribution and breaks, meetings and external activities are noise distributions.

5.2 Experiment Design

The two approaches for histogram based cleansing - slope based and kernel density estimate based approach (see Section 4) - are evaluated based on the

above workflow, in order to test their performance. Different combinations of mean processing, break, meeting, and external times are used as well as varying probabilities for the different states. Each experiment is executed several times and the means of the results are compared. Table 2 specifies the different parameters for each experiment. The processing times for the activities follow the Normal distribution in these tests, where $\text{procTime}(\mu) := \max(1, N(\mu, \mu))$. The five experiments are given in the next table. The processing times of activities are chosen in different sizes to investigate the effect on the estimate. Further, the probabilities of noise is varied, where the mean times of the noise has been chosen to be much smaller, equal to and much bigger than the processing times of activities. Scenarios 4 and 5 deviate most from a normal operation, since the user only works one fifth of her time on activities in the system.

Experiment	**Processing**					**Break**		**Meeting**		**External**	
	Act 1	Act 2	Act 3	Act 4	Prob	Mean	Prob	Mean	Prob	Mean	Prob
1	300	750	450	150	80%	20	10%	300	5%	4000	5%
2	300	750	450	150	50%	20	10%	300	5%	4000	35%
3	300	750	450	150	50%	20	10%	300	20%	4000	20%
4	300	750	450	150	20%	20	10%	300	35%	4000	35%
5	300	300	250	150	20%	20	10%	300	35%	4000	35%

Table 2. Summary of the different experiment parameters

5.3 Experiment Results

The results of the different experiments are summarized in Table 3. The table contains the experiment number as well as the processing time estimates for the different activities using the slope and the kernel density estimation (kde) based approach. Furthermore, columns with remarks and a filter disregarding activities with a duration smaller than the specified number of time units are included. The estimates significantly deviating from the processing times are highlighted in gray. From the table it results that all estimates for activity 3 and 4 are reliable and correct, thus, errors only happen for activity 1 and 2. Also, in all experiments the processing time estimate for activity 1 is significantly underestimating the processing time. Underestimation of the first activity of a workflow is a consequence of the start time estimation approach: since there is no preceding activity of the workflow, the best estimate is the closest completion time of another activity performed by the same user. Thus, as soon as the execution of activities interleaves the estimate will be significantly underestimated. Thus, the first activity in a workflow requires a special treatment, e.g., by adding a filter to cut off the significant underestimation. When applying a filter of at least 40 time units for a valid processing time, the correct results are derived for all experiments.

Experiment	Filter	Activity 1		Activity 2		Activity 3		Activity 4		Remark
		slope	kde	slope	kde	slope	kde	slope	kde	
1	0	9	25	734	742	446	448	149	148	input delay 500
	40	307	273	734	742	446	448	149	148	
2	0	14	8	11250	6564	451	452	148	146	input delay 500
	40	292	298	11250	6564	451	452	148	146	
	0	294	13	749	736	451	456	154	151	input delay 800
	40	294	297	749	737	451	455	154	150	
	0	302	304	738	749	438	445	151	151	input delay 1000
3	0	297	10	1017	6163	449	458	150	150	input delay 400
	40	297	296	1017	6163	449	457	150	150	
	0	297	10	1017	6163	449	458	150	150	input delay 600
	40	297	296	1017	6163	449	457	150	150	
	0	308	16	757	751	451	452	149	151	input delay 800
	40	308	301	757	752	451	452	149	151	
4	0	18	14	9115	7021	463	452	160	151	input delay 1000
	40	293	297	9115	7035	463	451	160	153	
	0	304	306	783	987	455	445	151	150	input delay 1500
	0	12	12	752	759	456	452	157	150	input delay 2000
	40	307	309	752	759	456	453	157	152	
5	0	15	15	4555	5187	267	407	152	151	run 1
		302	11	502	634	309	409	142	147	run 2
		303	302	304	298	252	248	156	150	run 3

Table 3. Estimates for the different experiments

The errors in activity 2 are due to significantly overestimating the processing time. The reason here for overestimation is a system overload. Since activities are queuing up, the chance of adding one or multiple times a noise contribution increases significantly. Therefore, noise can become a dominant distribution resulting in the overestimation. During normal operation this effect can not be observed and therefore the described approach provides good estimates. This statement is supported by the results of experiments 2, 3 and 4 which are executed with different mean delays between the triggering of subsequent process instances, thus influencing the system load. The lower the mean delay, the more the process instances are interwoven with each other, resulting in overestimating the processing time.

The three runs of experiment 5 with exactly the same settings give an indication that the actual estimation result for this particular experiment varies significantly depending on the actual executions of the workflows. While in the first run activity 1 is underestimated and activity 2 is overestimated, in the second run only the kde approach underestimates activity 1 and overestimates less drastically activity 2. Finally in the third run all estimates provide good results. This indifferent experiment result is due to the fact that this experiment is close to an overload situation and, depending on the assignment of tasks to users, an overload is observed or not. It should be noted that the other results given in

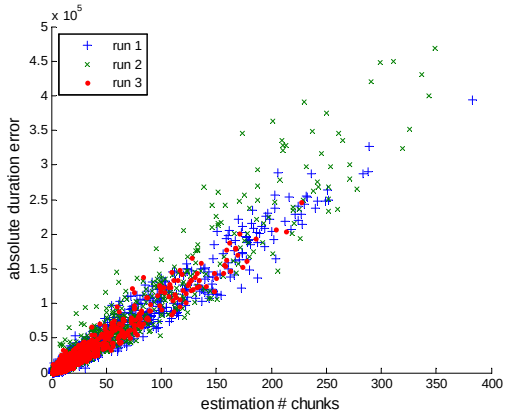

Fig. 11. Evaluation error estimate

Table 3 do not exhibit this behavior and are stable also when repeated several times.

Since overestimation of processing times can not be eliminated the aim is to find a measure of the quality of the processing time estimate of an activity. From the experiments we conclude that there is a correlation between the absolute error of the start time estimate and the number of chunks used in the start time estimate. Thus, the mean number of chunks and the corresponding standard deviation can be used as a quality indicator. Fig 11 depicts the scatter plot for activity 2 of experiment 5 for the three runs. Run 3 produces the best results indicated by most of the dots being in the lower left quadrant of the figure. Run 1 and 2 result in overestimations of the processing time and have data in the upper right quadrant of the figure. The correlation is observable since the cloud of dots approximates a diagonal. Since this graphical representation is hard to

Experiment	**Run**	**Activity 1**		**Activity 2**		**Activity 3**		**Activity 4**	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	1	1.0	0.0	48.3	54.6	18.9	23.2	26.4	35.4
5	2	1.0	0.0	46.5	59.5	23.9	30.3	15.3	22.0
5	3	1.0	0.0	24.4	34.7	22.4	31.2	20.3	26.0

Table 4. Mean and st. dev. of number of chunks in the estimated data for different runs of experiment 5

interpret, Table 4 contains the mean and standard deviation of the number of chunks in the estimated data for the three runs of experiment 5. The activity 2 column is based on the data depicted in Fig 11. The mean and standard deviation of activity 2 of run 3 are comparable to the means and standard deviations of

activities 3 and 4. The mean and standard deviation of activity 2 for run 1 and 2 are much higher, indicating the lower quality of the estimate. The measure is not applicable to activity 1 since the estimate for the start activity of a workflow will always depend on the completion of the previous activity by the same user. One can infer whether a mean and standard deviation combination give an indication of an unreliable processing time estimate by comparing them with the same measures of different activities and different workflows. An independent decision criterium is subject to future work.

6 Related Work

Several approaches on performance model mining are relevant as related work. Some are related to ProM [10] and are based on event logs provided in the Mining Extensible Markup Language (MXML) [11]. Rozinat et al. [11] present an approach to mine simulation models from MXML event logs. The idea is to generate a process model, represented as a Colored Petri Net (CPN). Depending on the event log's richness, the resulting CPN may cover not only the control-flow perspective, but also the resource and performance perspective. However, the essential difference between our approach and all the above-mentioned approaches, is that all of them assume the event log contains both start and end times of an activity, which is not the case in our scenario.

There is also some literature making less assumptions on the available event logs. For example, in [12] the authors try to derive the relation between events and process instance assuming there is no explicit data available to make the link. In [13] the authors address noisy event logs and ways of dealing with it. However, the focus there is not on performance models.

Classical performance models, such as, Queuing Networks [14] or stochastic Petri Nets [15] assume that the complete system is modeled. The models can then be used either to perform an equilibrium analysis or a transient analysis. In our situation the event log does not capture the complete system but only a part of it. To be able to apply classical performance models we have to make strong assumptions on the non-represented (parts of) the system(s).

It should be also noted that not all event logs are focusing on control flow performance mining. For example, in [16] the authors base their work on change logs, documenting ad-hoc changes performed on process instances. These change logs are then used to mine reference models.

7 Conclusion

In this paper we propose a systematic approach to prepare event log data from semi-structured processes. In particular, the main goal is to estimate duration distribution and the start time of an activity in the process. This is necessary, since in a semi-structured process, activities are not always performed solely in one computer system and therefore the start time of an activity cannot be acquired automatically. The resulting event log can then be further used in

combination with process mining techniques to actually infer a performance model. Thus, the main contribution of this paper consists of the mathematical formulation of the problem and the new approach for histogram based cleansing. Future work will strengthen the evaluation of our approach (using more complex processes), and will focus on better approximating the mixed model by assuming that noise distributions are constant for all activities. Also, since in this paper we have assumed that a user can only perform one activity at the time (which excludes parallel execution of activities), in future work we will investigate to what extent our approach should change in order to accommodate concurrency.

References

1. Rozinat, A., Mans, R.S., Song, M., Aalst, W.: Discovering simulation models. Information Systems **34** (2009) 305–327
2. Wombacher, A., Iacob, M., Haitsma, M.: Towards a performance estimate in semi-structured processes. In: Intl. Conf. on Service-Oriented Computing and Applications (SOCA). (2011) 1 –5
3. Tou, Gonzalez: Pattern Recognition Principles. Addison-Wesley, Reading (1974)
4. Parzen, E.: On the estimation of a probability density function and mode. Annals of Mathematical Statistics **33** (1962) 1065–1076
5. Rosenblatt, M.: Remarks on some nonparametric estimates of a density function. The Annals of Mathematical Statistics **1956** (1956) 832–837
6. Wombacher, A., Iacob, M.E.: Estimating the processing time of process instances in semi-structured processes - a case study. In: Service Computing (SCC). (2012)
7. Jensen, K., Kristensen, L., Wells, L.: Coloured petri nets and cpn tools for modelling and validation of concurrent systems. STTT **9** (2007) 213–254
8. Medeiros, A., Günther, C.W.: Process mining: Using CPN tools to create test logs for mining algorithms. In: Proc. of WS on the Practical Use of Coloured Petri Nets and CPN Tools (CPN). Volume 576 of DAIMI. (2005) 177–190
9. Jensen, K.: Coloured Petri Nets. Springer Verlag, Heidelberg (1992)
10. Dongen, B., Medeiros, A., Verbeek, H.M.W., Weijters, A.J.M.M., Aalst, W.: The proM framework: A new era in process mining tool support. In: Application and Theory of Petri Nets 2005, Springer (2005) 444–454
11. Rozinat, A., Mans, R.S., Song, M., van der Aalst, W.M.P.: Discovering colored petri nets from event logs. STTT **10** (2008) 57–74
12. Motahari-Nezhad, H., Saint-Paul, R., Casati, F., Benatallah, B.: Event correlation for process discovery from web service interaction logs. The VLDB Journal **20** (2011) 417–444
13. Musaraj, K., Yoshida, T., Daniel, F., Hacid, M.S., Casati, F., Benatallah, B.: Message correlation and web service protocol mining from inaccurate logs. In: IEEE International Conference on Web Services. (2010) 259–266
14. King, P.: Computer and Communication Systems Performance Modelling. Prentice Hall (1990)
15. Marsan, M.A.: Stochastic petri nets: an elementary introduction. In: Advances in Petri Nets. (1989) 1–29
16. Li, C., Reichert, M., Wombacher, A.: Discovering reference models by mining process variants using a heuristic approach. In: Business Process Management. Volume 5701 of LNCS. Springer Berlin / Heidelberg (2009) 344–362