Towards a Performance Estimate in Semi-Structured Processes

Andreas Wombacher
University of Twente,
Enschede, The Netherlands
Email: a.wombacher@utwente.nl

Maria Iacob
University of Twente,
Enschede, The Netherlands
Email: m.e.iacob@utwente.nl

Martin Haitsma
University of Twente,
Enschede, The Netherlands
Email: martin.haitsma@gmail.com

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 information required for a performance model 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 incremental cleansing of data. The proposed approach is evaluated in a case study where the quality of the derived event log is assessed by domain experts.

I. INTRODUCTION

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 can be found in scenarios where several people potentially from different organizations cooperate e.g. in creating a yearly progress report or writing a scientific paper. Other examples are workflows where people interact with clients and/or paper documents which are used to insert, approve, or validate information in a potentially Web based information system. These Web based information systems can be an application server or orchestrated services e.g., using BPEL.

Nevertheless, in these scenarios it is important for the management to better understand the process, the characteristics of activities, and the performance of individual employees. Lacking such knowledge makes it hard to predict the load of resources and to make a balanced resource planning. For example, it is difficult to predict the ability of the business to handle higher workload due, for example, to a promotion activity or to vacations.

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 or when an employee starts to read a paper request form of a client. 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 this paper we aim to use the available log information to perform data analysis and data cleansing in order to get an estimate of the starting time, from which the underlying performance model can be further inferred. Thus, we propose a structured approach to investigate and cleanse the observed event data. The result is an estimated starting time for each event. In case the estimated starting time is not trustworthy we report it as 'unknown'.

II. USE CASE

The proposed approach has been motivated and evaluated on a real-life use case. Due to a non-disclosure agreement the labels of activities have been made more generic and no absolute performance data is provided. The use case is the processes in the front-office of a service provider for a financial company. The service provider uses a web service-based application to quickly set up 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), handling the contracting, and sending the application to the back office for further handling.

The framework provides a proprietary process modeling language which is based on states, and manual and automatic state changes, performed respectively by an employee or by the software. The expressiveness of the modeling language is comparable to that of Finite State Automata, thus supports loops but no parallelism. Due to the processes at hand, the system only documents the completion of a state change (activity), and thus not the starting of an activity.

The data used in the use case have been collected from the end of September 2010 until mid February 2011. It should be noted that users spend only part of their time working in this system. However, we can state that the average number of hours per user spend working in the framework system stays approximately the same over the investigated period of time.

Figure 1. Start time inference

III. PROBLEM DESCRIPTION

The challenge posed with semi-structured processes is that start times of activities cannot automatically be recorded by the underlying system. Another challenge is that users often work on more than one process instance and therefore the percentage of time a user is working on the process instance under investigation is unknown. Further, 'internal' activities like e.g., meetings, coffee breaks, early departure of an employee are not documented and therefore are not available for the start time estimation.

After estimating a start time, the derived performance model has to be applied carefully. Since employees work on more than one process instance of which no performance model is available, it is impossible to make statements about how fast the incoming requests can be processed. However, an estimate of how many hours the employees have to spend on the process to handle these requests can be determined. This is valuable information for the management, which should have an overview of the workload caused by other processes.

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, the source and the target state. 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 visualized in Fig 1.

In the following we assume that the process is involving potentially multiple systems each providing part of the log information. However, we are not addressing neither data integration problems such as entity resolution problems of event log information nor syntactic or semantic data integration problems.

IV. APPROACH

The approach presented here is based on the steps depicted in Fig 2. A first cleansing step is performed on the raw event data. Next the cleansed data is used to infer an initial estimate of the start time for each activity. The initial start time estimates may be overwritten in later cleansing steps. The following cleansing step investigates special situations per process instance (also called case). The last cleansing step is the histogram based cleansing removing outliers, i.e., exceptionally high durations of activities. The final step

investigates dependencies of activity durations cross process instances and categorical data like, e.g., the weekday or the experience of a user. Thus, the final step tries to verify 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 using existing tools. Due to lack of space only a high level view can be provided. More details can be found in [2]

A. Raw Event Data Cleansing

The initial step of the data cleansing is to make sure that the basic characterization as given in Sect III actually applies to the event log data. In particular, we are checking whether the partial order of the Completion Time and the complete order of the State Change ID are not conflicting with each other. An inconsistency of the two orders can be caused by the fact that the Completion Time of an activity is determined at a different point in time than the moment when the number representing the State Change ID is assigned. This effect is e.g. caused by executing the workflow in a distributed infrastructure or by performing external service invocations. It is important to have a complete order, thus, a new complete order has to be defined based on the available orders. We keep the inconsistent orders since the fact that there are inconsistencies is important information for further cleansing steps.

The second step of the cleansing aims to ensure the reliability of the data, thus, establishes whether the data at hand reflects normal operation of the system or an exceptional mode of operation. An example of an exceptional mode of operation are network problems in a distributed infrastructure. These errors are often related to the unavailability of components or services, such as, external services, the logging server or the network. Furthermore, infrastructure problems observed during a timespan influence the events related to various cases. Consequently, the only option to clean the data is to exclude the data collected during the identified time span. Potentially more fine grained exclusion criteria can be defined, but this depends on the actual workflows and the used infrastructure. In general infrastructure problems may result in the event log in incorrect ordering of state changes, missing state changes, or duplicate state changes.

B. Start Time Estimate

Estimating the start time of an activity is based on a complete order of state changes (activities), which is consistent

Figure 2. Cleansing rules overview

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 1 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 execution time of 25 minutes and 6 seconds as depicted in Fig 1.

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 1 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 execution time of 5 minutes and 40 seconds as depicted in Fig 1.

Thus, the estimated start time of an activity is the maximum of (i) the completion time of the preceding activity of the same process, and (ii) the completion time of the preceding activity of the same user.

C. Process Instance based Cleansing

The third step investigates the event log per process instance, also called case, and marks complete cases as unsuitable for performance model mining. In particular, we are considering special test cases performed on the system, as well as special deadlock and livelock errors.

1) *Test cases*: Productive systems undergo an evolution over time, thus hardware and software updates are performed. To ensure the reliable operation of the software, i.e., the implemented processes, it is necessary to perform tests. Test data should be excluded from the event log. To exclude the test cases from the event log criteria have to be determined to identify activities in the event log to be part of a test case.

2) *Deadlock state changes*: Due to a bug in the code or any other error it can happen that a process case is blocked in a state (i.e., endlessly waiting for the exit criteria). In that case a user with admin rights can manually perform a state change, ignoring the exit criteria. Ideally, the transitions which are executed ignoring the criteria should be flagged, such that, these can easily be excluded in the generation of the performance model. If this is not the case, these state changes have to be filtered out based on a determined criterium. This can be done manually by asking the administrator which transitions were performed outside the normal flow. Another way is to extract the business rules and then

exclude the state changes which do not conform to these rules.

3) *Livelock state changes*: A livelock is similar to a deadlock, except that the process continuously performs state changes but is unable to complete the process, i.e., the process execution cannot leave a loop.

Livelocks can be detected by counting the repetitions of a certain transition. If the count is above a certain threshold (e.g., five repetitions), the system should give an alert to fix this error. If the system does not have such functionality, livelocks can be treated similar to deadlocks, since they must be resolved through the intervention of an admin user by resolving infrastructure problems or by manually performing a state change again. Since livelocks are exceptional situations, the corresponding cases must be excluded.

D. Histogram based Cleansing

Based on the remaining process instances in the event log, the next step is to investigate the histograms of activity durations with the same label over all process instances. The duration is defined as the difference between the Completion Time of an activity and its estimated start time. Based on the histogram a threshold can be defined, i.e., when a duration is considered a too strong deviation from expectations. For these activities, the start time is set to *unknown* and these activities are not further considered. Two examples for strong deviations are:

1) *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.

2) *Non-visible Activities*: In the proposed approach we 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, (e.g. attending a meeting, having a coffee break or lunch, or working in a different system) non visible activities, since they are activities of the user, but they are not documented in the event log. The threshold for the extreme values can be determined either by a percentile score (e.g., the upper 10 percent of the values), by z-score (e.g., more than two standard deviations above the average), or by domain experts.

To illustrate the effect, we consider the 'send request' activity of the use case workflow as discussed in Sect II. In total there are 4774 executions of this activity remaining

Figure 3. Histogram of the Duration of Activity 'new request'

Figure 4. Visualization of the Weekday Probability Distribution in Percent

in the dataset.¹ The related histogram of the durations of this activity is depicted in Fig 3. The average duration of the activity is 229 sec and the standard deviation is 319 sec indicating that the deviation of the data is quite big. Applying a threshold of a percentile of 10% on the data, means that all durations longer than 510 sec are neglected, which excludes 490 activities in the data set.

E. Data Independence Test

The last step is to perform an analysis of the independence assumption of the data. In a performance model the assumption is that executing an activity follows always the same distribution independent of the day of the week, the experience of the user, or the user itself. Since all the characteristics mentioned are categorical data, we propose to perform a χ^2 test for homogeneity[3]. The aim is to determine whether the distributions of durations observed in each category can be considered as having the same distribution. A basic requirement of the approach is that more than 80% of the durations contain at least 5 observations.

1) *Weekday independence*: The analysis for the weekday is based on data visualized in Fig 4 for activity 'new request' based on the cleansed data. The numbers provided represent the duration distribution as a percentage of the overall

¹It should be noted that these are numbers which can be directly mapped to the actual number, but are not the real numbers.

number of executions for a particular weekday. Percentages are used instead of absolute numbers since the variations in the absolute number per weekday were so high that the test would not provide reliable results. The null hypothesis that the same distribution applies for all categories can be verified if the calculated value Q is below $\chi^2_{df;\alpha}$, which is the α quantile of the χ^2 distribution for a degree of freedom df .

Applying these formulas to the data presented above for the degree of freedom df is 60 and the 99% quantile of the χ^2 distribution shows that the distributions observed per weekday are considered to be based on the same distribution. Thus, the observed durations are independent of a particular weekday.

2) *Iteration independence*: A process may contain cycles/loops. It has to be checked whether the durations per iteration are equally distributed, i.e., whether the second iteration generally takes less time than the first one. We apply the same approach to the data for the 'new request' activity indicating that the first and the subsequent iterations do not follow the same distribution.

In the example process, the activity 'New Request' can be repeated multiple times for one process case. In the first iteration, the full client data has to be obtained, while in the later iterations only partial information has to be adapted. A second iteration is required in case an error has to be resolved or the back office needs additional information. This second or further iterations take significantly less time than the first one. As a consequence the iterations of activities have to be distinguished for creating a performance model.

3) *Discussion*: Having data independence is a critical requirement for determining a performance model. In case data dependency is concluded a possible solution is to resolve these dependencies by further distinguishing activities. In the loop scenario, a possibility would be to classify the 'new request' activity as contained in the original event log into two activities: 'first new request' and 'repeated new request' activity. Based on this distinction data independence can be confirmed and a performance model can be derived.

In situations where a refinement of activities is applied, the histogram based cleansing and the data independence test have to be repeated to determine a cleansed event log, which can be used to mine a performance model.

V. EVALUATION

The result of the approach presented in this paper is a cleansed event log, in particular the start time estimates which will also be the focus of evaluation.

The aim of the evaluation is to see whether the performance model per activity which can be directly derived from the cleansed event log, conforms to the expectations of the managers in the company. Since there is no performance model available at the company, we made a questionnaire for

the analyst in the bank and the analyst of the software supplier to estimate the durations for activities of a process. The time-estimates follow the Project Evaluation and Review Technique (PERT) [4]. The idea is that a domain expert gives three time estimates for each activity; an optimistic estimate, or the minimum time in the most favorable conditions, a pessimistic and the most likely time. The expected time for each activity is a weighted average of these estimates, following the formula (optimistic time + 2x average time + pessimistic time)/4.

This assessment has been performed for several activities. The conclusion is that the data in the cleansed log file is indeed in the range of the expected durations. In case of the 'new request activity' the durations contained in the log file is in average about 4 minutes while the optimistic estimate of the experts has been 3 and 5 minutes. Adding the standard deviation observed in the log file, we get around 10 minutes which is the estimated average. The challenge with the PERT method is that there is an assumption made on the underlying distribution, which may deviate in the actually observed distribution.

Over all activities investigated it turns out that the bank analyst is more optimistic with his estimates and as a consequence is closer to the estimates contained in the cleansed event log. We presented the results of this study to the experts and they found the discrepancy with the estimates contained in the log file explainable. Overall they were content with the accuracy of the results.

VI. RELATED WORK

There is quite some related work on performance model mining. Many approaches have been implemented in the context of ProM [5] assuming that the event log contains the start and end time of an activity, which is not the case in our scenario.

However, there is also some literature making less assumptions on the available event logs. For example, in [6] the authors try to derive the relation between events and process instance assuming there is no explicit data available to make the link. In [7] 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 [8] assume that the complete system is modeled. In our situation the event log does not capture the complete system but only a part.

Not all event logs are focusing on performance of control flow mining. For example, in [9] the authors base their work on change logs, i.e., documenting ad-hoc changes performed on process instances. These change logs are then used to mine reference models.

VII. CONCLUSION

In this paper we propose a systematic approach to prepare event log data from semi-structured processes for the

derivation of a performance model. In particular, the main goal is to estimate 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 start time estimates are checked for outliers based on various errors and the independence of situational characteristics is checked.

REFERENCES

- [1] A. Rozinat, R. S. Mans, M. Song, and W. Aalst, "Discovering simulation models," *Information Systems*, vol. 34, no. 3, pp. 305–327, 2009.
- [2] A. Wombacher, M. Iacob, and M. Haitisma, "Towards a performance estimate in semi-structured processes," CTIT Technical Report, Tech. Rep., 2011.
- [3] J. Lehn and H. Wegmann, *Einfuehrung in die Statistik*, 2nd ed. Teubner, 1992.
- [4] K. M. van Hee and H. A. Reijers, "Using formal analysis techniques in business process redesign," in *Business Process Management*. Springer, 2000, pp. 142–160.
- [5] B. Dongen, A. Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. Aalst, "The proM framework: A new era in process mining tool support," in *Application and Theory of Petri Nets 2005*. Springer, 2005, pp. 444–454.
- [6] H. Motahari-Nezhad, R. Saint-Paul, F. Casati, and B. Benatallah, "Event correlation for process discovery from web service interaction logs," *The VLDB Journal*, vol. 20, pp. 417–444, 2011.
- [7] K. Musaraj, T. Yoshida, F. Daniel, M.-S. Hacid, F. Casati, and B. Benatallah, "Message correlation and web service protocol mining from inaccurate logs," in *IEEE International Conference on Web Services*, 2010, pp. 259–266.
- [8] P. King, *Computer and Communication Systems Performance Modelling*. Prentice Hall, 1990.
- [9] C. Li, M. Reichert, and A. Wombacher, "Discovering reference models by mining process variants using a heuristic approach," in *Business Process Management*, ser. LNCS. Springer Berlin / Heidelberg, 2009, vol. 5701, pp. 344–362.