Process Mining Applied to the BPI Challenge 2012: Divide and Conquer While Discerning Resources

R.P. Jagadeesh Chandra Bose and Wil M.P. van der Aalst

Department of Mathematics and Computer Science
Eindhoven University of Technology, The Netherlands
{j.c.b.rantham.prabhakara,w.m.p.v.d.aalst}@tue.nl

Abstract. A real-life event log, taken from a Dutch financial institute, is analyzed using state-of-the-art process mining techniques. The log contains events related to loan/overdraft applications of customers. We propose a hierarchical decomposition of the log into homogenous subsets of cases based on characteristics such as the final decision, offer, and suspicion of fraud. These subsets are used to uncover interesting insights. The event log in its entirety and the homogeneous subsets are analyzed using various process mining techniques. In this paper, we present results related to (a) resource perspective and their influence on execution/turnaround times of activities, (b) control-flow perspective, and (c) process diagnostics. A dedicated ProM[1] plug-in developed for this challenge allows for a comprehensive analysis of the resource perspective. For the analysis of control-flow and process diagnostics, we use recent, but pre-existing, ProM plug-ins. As the evaluation shows, our mix of techniques is able to uncover many interesting findings and could be used to improve the underlying loan/overdraft application handling process.

1 Dissecting the Event Log

The event log used for the BPI Challenge 2012 contains events related to the application process for a personal loan or overdraft within a Dutch financial institute. The event log contains 13,087 cases and 262,200 events distributed over 36 activities having timestamps in the period from 1-Oct-2011 to 14-Mar-2012. The global process is defined over three sub-processes and can be summarized as follows: a submitted loan/overdraft application is subjected to some automatic checks. The application can be declined if it does not pass any checks. Often additional information is obtained by contacting the customer by phone. Offers are sent to eligible applicants and their responses are assessed. Applicants are contacted further for incomplete/missing information. The application is subsequently subjected to a final assessment upon which the application is either approved and activated, declined, or cancelled. The different categories of loan/overdraft applications can be classified according to the tree illustrated in Fig. 1.

[1] ProM is an extensible framework that provides a comprehensive set of tools/plug-ins for the discovery and analysis of process models from event logs. See http://www.processmining.org for more information and to download ProM.

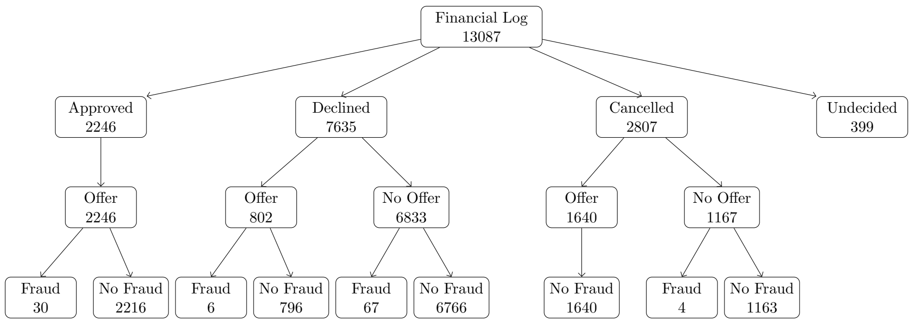


Fig. 1. Classification of applications based on their assessment.

Out of the 13,087 cases, 2246 cases were approved while 7635 cases were declined and 2807 cases were cancelled. 399 cases were still running and no decision was taken yet (class "undecided" in Fig. 1). For all cases that have been approved, an offer was made to the applicant. However, as Fig. 1 shows, cases can be declined or cancelled without making an offer. For example, 6833 cases were declined without making any offer while 802 cases were declined after making an offer. Furthermore, certain cases are considered suspicious and a special kind of check (W_Beoordelen fraude) was performed. Cases for which this check was executed are classified as "Fraud"; all other cases are classified as "No Fraud". For example, 30 of the 2246 cases that were approved are considered suspicious (class "Fraud"). As can be seen from Fig. 1, the percentage of suspicious cases is negligible ($\approx 0.85\%$).

Each case in the event log has an attribute AMOUNT_REQ, which specifies the amount of loan/overdraft requested by a customer (called claim amount in the remainder). We divide the claim amounts in bins of 10,000 units and filter (remove) all undecided cases and bins where the number of cases is less than 100. This results in 12,613 cases. Table 1 depicts the distribution of cases in each bin according to their final application assessment, i.e., approved, declined, or cancelled. It can be seen that the cases with claim amounts between 0 and 10,000 and between 50,000 and 60,000 have a relatively low rate of approval (12.65% and 14.24%) when compared to the rest, which is between 20.95% and 24.66). Similarly, relatively many cases in these bins are declined.

Assessing the Quality of Data

Before we perform a rigorous analysis of the event log, we first assess some quality aspects of the log.

– *Missing associations between start and complete events of activities:* For activities that have both the start and complete event types, we check for each activity if every instance of complete event type has a corresponding

Table 1. Claim Amount and Final Assessment of Cases

Claim Amount	# Cases	A_Approved	A_Declined	A_Cancelled
$0 - 10,000$	6095	771 (12.65%)	4197 (68.86%)	1127 (18.50%)
$10,001 - 20,000$	3627	808 (22.28%)	1928 (53.16%)	891 (24.57%)
$20,001 - 30,000$	1593	389 (24.42%)	776 (48.71%)	428 (26.87%)
$30,001 - 40,000$	665	164 (24.66%)	324 (48.72%)	177 (26.62%)
$40,001 - 50,000$	296	62 (20.95%)	158 (53.38%)	76 (25.68%)
$50,001 - 60,000$	337	48 (14.24%)	206 (61.13%)	83 (24.63%)

start event type and vice versa. In the event log, we have six activities that have both the start and complete event types. Table 2 depicts the number of instances of such missing associations in the event log and the number of traces where they manifest. For example, one instance of W_Afhandelen leads-complete has a missing W_Afhandelen leads-start event in one of the traces. For each trace, there is just one missing association per activity. Overall, in the event log, there are 1042 traces where an association is missed for one of these activities.

Table 2. Missing associations between start and complete instances of activities

Activity	Missing Start		Missing Complete	
	# Traces	# Activity Instances	# Traces	# Activity Instances
W_Afhandelen leads	1	1	0	0
W_Beoordelen fraude	0	0	0	0
W_Completeren aanvraag	455	455	0	0
W_Nabellen incomplete dossiers	571	571	1	1
W_Nabellen offertes	6	6	2	2
W_Valideren aanvraag	7	7	0	0

– *Overlapping instances of activity executions:* For activities that have both the start and complete event types, we analyze if there are any scenarios where activity executions overlap while using the same resource. This corresponds to pairs of instances of the start of an activity before its completion by the *same resource* leading to ambiguities in associating a complete event with its corresponding start event. Fig. 2 depicts such a scenario where there is an ambiguity in associating the instances of activity $\mathsf{a}_{complete}$ to instances of activity a_{start}. It is not possible to uncover whether the first occurrence of $\mathtt{a}_{complete}$ corresponds to the first or the second occurrence of $\mathtt{a}_{start}$.[2] Fortunately, only one trace (trace name: 212836) exhibits this behavior in the event log.

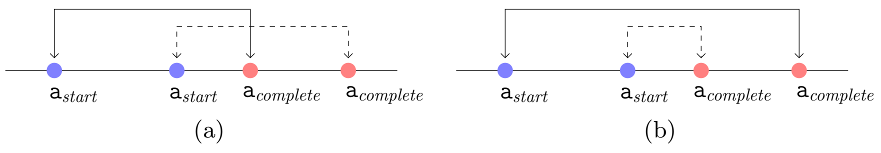


Fig. 2. Ambiguities in associating events related to multiple instances.

– *Missing resource information:* There are some events in the log where the resource information is missing. Overall, there are 18009 events across 3528 traces that have missing resource information, i.e., 6.86% of events and 26.96% of the traces have partially missing resource information.

2 Analyzing the Resource Perspective

In this section, we focus on the resource perspective and analyze whether there are remarkable differences between resources. Understanding the correlations between resources, workloads, and processing speeds of cases is gaining attention in recent years in process mining [1, 2].

The event log contains 69 distinct resources who have worked on at least one case in the log. Fig. 3 depicts the number of resources working per day during the time period covered by the event log. A resource is considered to be working on the process on a day if he/she is associated with at least one activity executed on that day. We can see that although the log contains 69 resources, on average only 20-30 resources work on the process on any given day. As expected, fewer resources work during weekends as is clearly shown by the pattern in Fig. 3. It is interesting to see that on Mondays, more number of resources are deployed. This can be attributed to the fact that there is a backlog of cases arriving on weekends, which can be tackled by deploying more resources. The number of resources working on the process from Monday to Friday follows a U-shaped pattern, with the number of resources being the minimum on Wednesdays. More number of resources are deployed on Fridays as well to reduce the backlogs. Certain resources work only on specific days (e.g., resources 10789, 10862 work only on Mondays and/or Fridays) on this process. Fig. 4 depicts the histogram of the number of resources and their total number of working days within the time period of the event log. We can see that around 18 resources

[2] Note that if the two instances of $\mathtt{a}$ are executed by different resources, it would have been possible to disambiguate.

work less than 20 days over the time period. Resource 112 works on each and every day (as can be seen in the histogram in the far right corner). In addition, resource 112 works throughout the day (24 hours). *This makes us believe that resource 112 is an automated resource.*

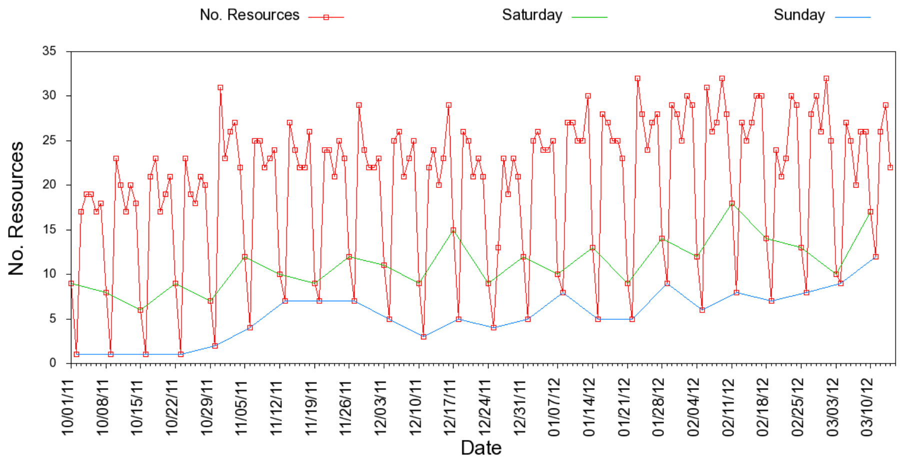


Fig. 3. Number of resources working on the cases per day

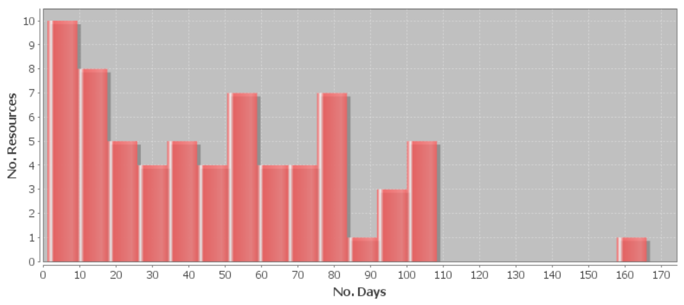


Fig. 4. Histogram of number of working days for resources

2.1 Resource–Application Assessment Task Matrix

Next, we investigate the relation between the resource handling the application and the final outcome. Table 3 depicts the number of cases assessed by resources based on their final assessment. We have considered only those resources who have been involved in assessing at least 250 applications. We observe that only selective resources[3] have the privilege of approving a loan/overdraft. We can see from Table 3 that resource 112, which corresponds to an automated resource, is involved in the approval of 3 applications. *The low number and the fact that 112 is an automated resource, make these three cases interesting from an auditing point of view. They are outliers and should be investigated in more detail.*

Table 3. Frequency of cases classified according to their final assessment for resources.

Resource	# Approved	# Cancelled	# Declined
112	$\boxed{3}$	1004	3429
10609	335	5	206
10138	681	5	156
10809	271	1	87
10629	359	1	119
10972	518	3	106
10910	0	23	244
11169	0	63	238

2.2 Resource–Activity Execution Time Analysis

In this section, we analyze the execution and turnaround times spent by resources on different activities. We need to differentiate between two types of activities. Activities that have the start and complete event types and activities that have only the complete event type. The former class of activities enables the computation of execution times while the latter class of activities are assumed (in this paper) to be atomic executions.[4] Considering the time difference between the activity completion and start timestamps as a notion of *execution time* of an instance of the activity is incorrect due to the fact that resources can be executing multiple activities simultaneously. We consider this time difference to be the *turnaround time* rather than the pure *execution time*. In order to estimate the execution times spent by resources on different activities, we need to consider the notion of working slots.

A working slot for a resource r is defined to be a time interval $[w_s, w_e]$ such that the resource r has started the execution of an activity at w_s and completed the execution of the activity at w_e (for atomic tasks, $w_s = w_e$). A special consideration needs to be done for activity executions that cover over multiple days.

[3] Nine resources are involved in approving a loan/overdraft application in the event log. The resources correspond to 112, 10138, 10609, 10629, 10779, 10809, 10972, 11289, and 11339

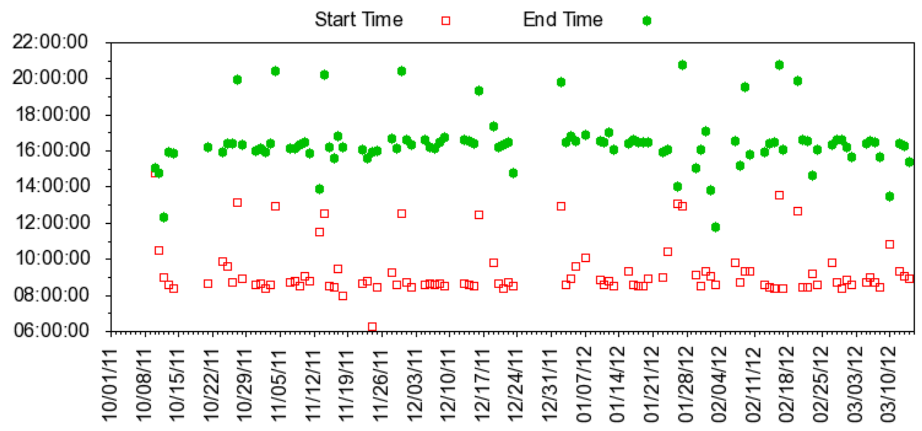
[4] One could use heuristics when considering the duration of such activities, e.g., assuming a fully sequential process, we can consider the execution time of an activity to be the time difference between the completion of the activity and its predecessor.

It could be the case that a resource starts the execution of an activity on some day but finishes its execution on a different day. We create two working slots for such activity executions. Let w_s and w_e be the start and complete timestamps of the execution of an activity and let d_1 be the day when the activity was started and let d_2 be the day when the activity execution was completed. We create one working slot $[w_s, d_1^e]$ and another working slot $[d_2^s, w_e]$ where d_1^e is the closing time of the resource on d_1 and d_2^s is the starting time of the resource on d_2. The starting and closing times of a resource on any given day can either be taken to be the first and the last timestamp logged by that resource on that day or the average starting and closing time of the resource. Fig. 5(a) and Fig. 5(b) depicts the starting and closing working times of resources 10138 and 11119 respectively across different days. We can see that resource 10138 on average starts working around 8:30 in the morning and closes at around 16:30 in the afternoon. On some occasions, the resource works in a different shift between 13:00 and 21:00. In contrast, resource 11119 works on average between 17:00 and 21:00.

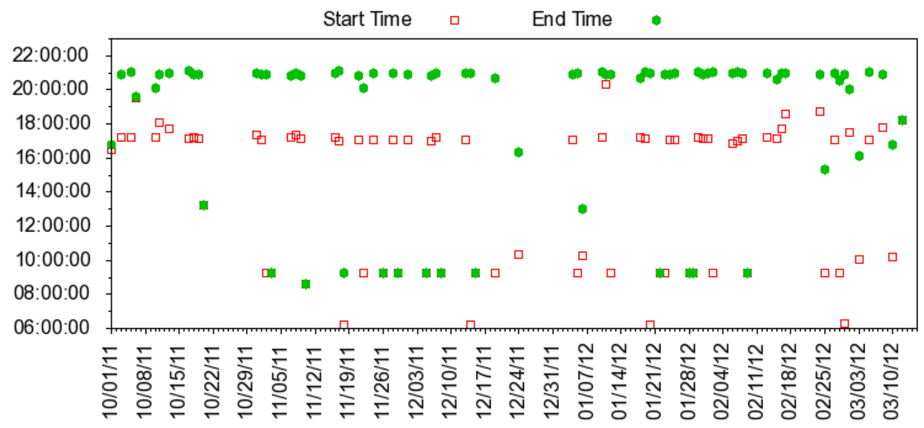
Fig. 6(a) depicts a generic scenario of working slots of a resource on a given day while Fig. 6(b) represents the same working slots as in (a) but marked with indices. We can see that working slots can overlap in different ways. For example, working slot $\boxed{3}$ starts during working slot $\boxed{2}$ while the working slots $\boxed{6}$ and $\boxed{7}$ are completely subsumed in working slot $\boxed{5}$. As another example, working slot $\boxed{9}$ starts in working slot $\boxed{8}$ and ends in working slot $\boxed{10}$. It is important to note that these working slots may correspond to activity executions for different cases, but all are executed by the same resource.

In order to deal with activities that are simultaneously executed, we consider interval slots. Informally, a working day for each resource is partitioned into a set of interval slots I. Each interval $i \in I$ is characterized by a start time i_s and an end time i_e. Furthermore, every start time, w_s, of a working slot w performed by that resource on that day corresponds to an interval start time, i_s, of some interval $i \in I$ and every end time, w_e, of a working slot w performed by that resource on that day corresponds to an interval end time, i_e, of some interval $i \in I$. Fig. 7 depicts the partitioning of a day into interval slots for the working slots depicted in Fig. 6. In any interval slot, there could be zero or more working slots. For example, no working slots exist in the interval slots {②, ⑥, ⑧, ⑭} while there are three working slots (corresponding to working slots $\boxed{5}$, $\boxed{6}$, and $\boxed{7}$ cf. Fig. 6(b)) in the interval slot ⑪. Similarly, each working slot is divided into a set of interval slots. For example, working slot $\boxed{5}$ (cf. Fig. 6(b)) is partitioned into the interval slots {⑨, ⑩, ⑪, ⑫, ⑬} while working slot $\boxed{7}$ corresponds to only one interval slot {⑪}.

Having defined the relationship between interval slots and working slots, we can now estimate the activity execution times. *We assume that the duration of an interval slot is uniformly utilized by all the working slots (and thereby the*



(a) Resource 10138



(b) Resource 11119

Fig. 5. Staring and closing working times of resources on different days.

activities) in that interval slot. Let $W_{r,d}$ be the set of working slots for resource r on day d. Let $I_{r,d}$ be the corresponding set of interval slots for resource r on day d. Let $\hat{w} \in W_{r,d}$ be the working slot pertaining to an instance $\hat{t}$ of activity t. Let $f_{r,d} : I_{r,d} \rightarrow 2^{W_{r,d}}$ be a function defining the set of working slots for an interval slot $i \in I_{r,d}$ and let $g_{r,d} : W_{r,d} \rightarrow 2^{I_{r,d}}$ be a function defining the set of interval slots for a working slot $w \in W_{r,d}$. The execution time of $\hat{t}$ is defined as:

$$ExecutionTime(\hat{t}) = \sum_{i \in g_{r,d}(\hat{w})} \frac{(i_e - i_s)}{|f_{r,d}(i)|}$$

In other words, the execution time of an activity instance pertaining to a working slot is the sum of normalized duration of the interval slots defined by that

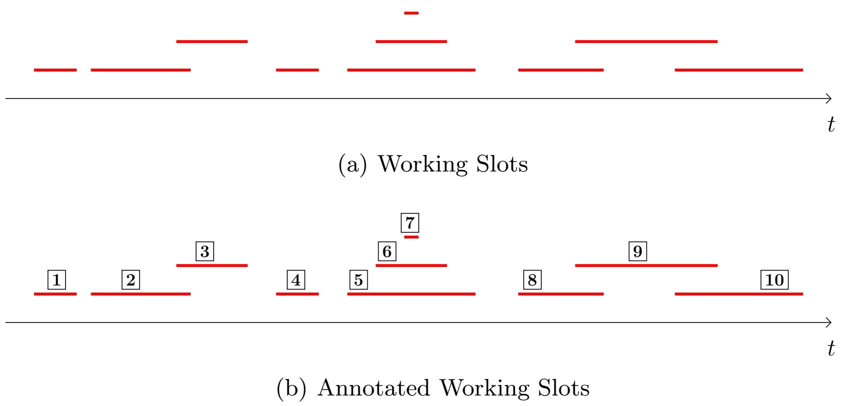


Fig. 6. Illustration of working slots of a resource in a day.

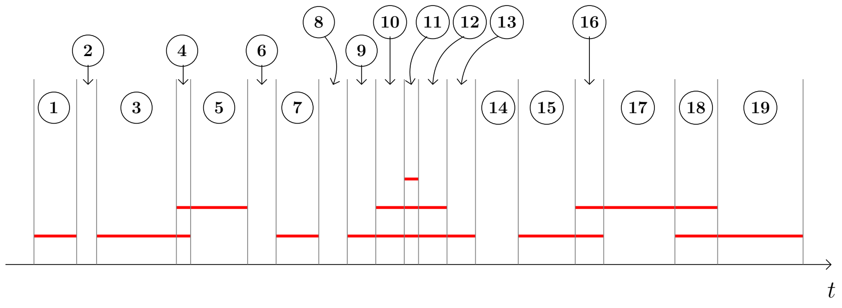


Fig. 7. Interval slots corresponding to the working slots defined in Fig. 6.

working slot. Consider working slot $\boxed{5}$ in Fig. 6(b). Let $\hat{t}$ be the activity instance corresponding to this working slot. The set of interval slots corresponding to this working slot is $g_{r,d}(\boxed{5}) = \{⑨, ⑩, ⑪, ⑫, ⑬\}$. In interval slot ⑨ only $\hat{t}$ is executed, so interval slot ⑨ contributes wholly to the activity execution time of $\hat{t}$. In interval slot ⑩, two activities (working slots $\boxed{5}$ and $\boxed{6}$) are executed. Therefore, the duration of interval slot ⑩ is normalized by 2 and assigned to the execution time of $\hat{t}$. In interval slot ⑪, three activities are simultaneously executed (working slots $\boxed{5}$, $\boxed{6}$, and $\boxed{7}$); so the duration of interval slot ⑪ is normalized by 3 and assigned to the execution time of $\hat{t}$. Hence, the total activity execution time of $\hat{t}$ is

$$\mathit{Execution\ \ Time}(\hat{t}) = \frac{[\![⑨]\!]}{1} + \frac{[\![⑩]\!]}{2} + \frac{[\![⑪]\!]}{3} + \frac{[\![⑫]\!]}{2} + \frac{[\![⑬]\!]}{1}$$

where $[\![\textcircled{i}]\!]$ is the duration of interval slot $\textcircled{i}$.

The turnaround time of $\hat{t}$ is defined as the length of the working slot, i.e., $Turnaround\ Time(\hat{t}) = (\hat{w}_e - \hat{w}_s)$.

We have analyzed the event log to see if there are perceivable differences in the average execution times of activities between different resources. We have noticed that certain resources tend to work on multiple loan/overdraft applications simultaneously while others tend to work on just one application at a time. For example, Fig. 8 depicts the working slots for resource 11002 on 09-Jan-2012. The x-axis represents the time and the y-axis represents the traces on which the resource works on that date. Each bar in the figure from (x_1, y) to (x_2, y) depicts one or more working slots covering time period $[x_1, x_2]$ for some activity(ies) in trace y.[5] The resource works in the evening shift and starts at around 17:30 hrs. The resource performs some tasks for trace 191644 between 17:35 and 17:50 hrs. Subsequently, the resource performs no work on this process until 18:30 hrs. The resource then starts to work on trace 190627 and after a couple of minutes also starts working on another trace 199324 simultaneously. Similarly, the lone working slot corresponding to the trace 190842 is subsumed in the first working slot for the trace 191797 while the working slot for trace 199381 is overlapping with the second working slot of 191797.

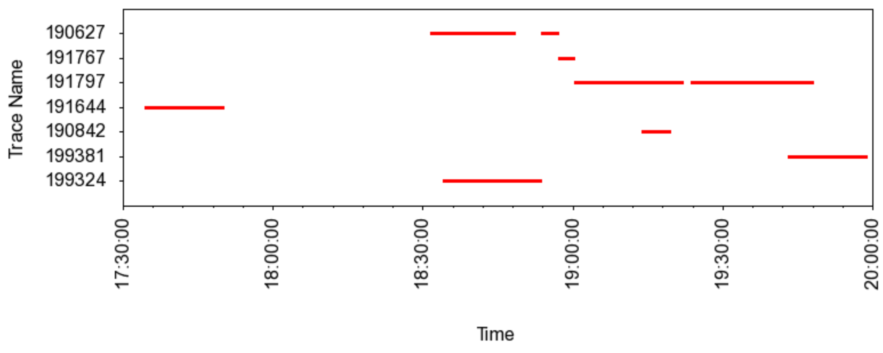
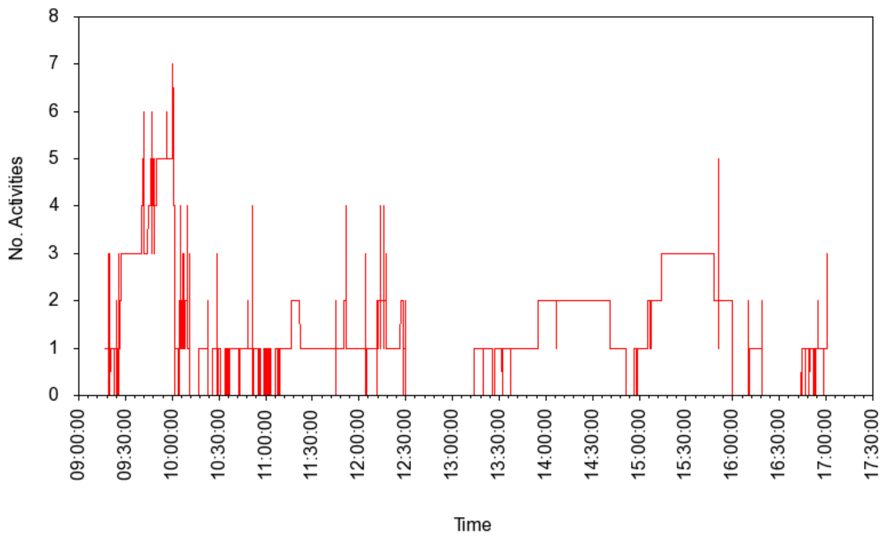


Fig. 8. Working slots for resource 11002 on 09-Jan-2012.

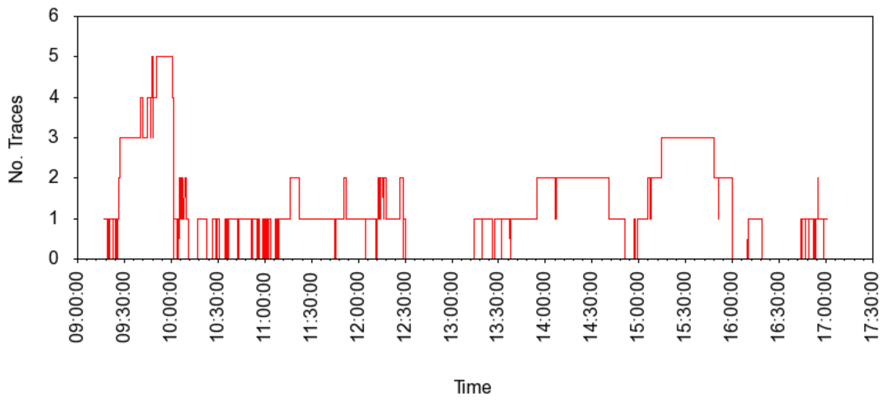
Fig. 9(a) depicts the number of activities worked upon by resource 11169 on 31-01-2012 at different periods of time while Fig. 9(b) depicts the number of traces that the resource was working on at different periods of time. From the figure we can see that this resource was working on two to five loan applications

[5] overlapping, subsuming, and adjacent working slots for activities in the same trace that fall in the interval $[x_1, x_2]$ are captured in the same bar in the figure.

simultaneously between 09:30 and 10:00 hrs and between two to three applications between 15:06 and 16:00 hrs. We can further see that the resource was idle between 12:30 hrs and 13:15 hrs (this might most likely correspond to a lunch break).



(a) Simultaneous activity executions



(b) Simultaneous trace executions

Fig. 9. Number of simultaneous activities and traces that resource 11169 worked on 31-01-2012

.

Table 4 depicts the top five resources who have worked on this process for at least

30 days and spent a significant amount of their working time processing multiple loan/overdraft applications. For example, resource 11169 worked on this process for 594.47 hours spread across 85 days. Out of this 594.47 hours, the resource spent 128.12 hours executing multiple loan applications simultaneously, which contributes to 21.55% of his working time. *Resources that are multitasking (i.e., simultaneously working on multiple cases) may have a negative effect on execution and turnaround times.* We see evidences of this effect in the event log. High execution/turnaround times might not be desirable by both by the customers as well as the organization.

Table 4. Top 5 resources who work on multiple loan applications simultaneously

Resource	No. Working Days	Total Working Time (hrs)	Simul. Working Time (hrs)	Perct. Simul. Working Time
11002	41	151.10	33.70	22.30%
11169	85	594.47	128.12	21.55%
10932	69	443.86	52.64	11.86%
11121	45	198.11	22.93	11.58%
10910	40	219.03	22.34	10.20%

Table 5 depicts the average execution and turnaround times for different activities (that have both start and complete event types) for resources 11169 and 11181. Both resources have comparable workload, i.e., the resources are comparable. While resource 11169 spends 21.55% of his/her working time executing multiple applications, resource 11181 spends only 2.62% of his/her working time on simultaneous applications. From the table we can see that the average execution and turnaround times for all the activities for resource 11169 is much higher (between 1.7 and 7.3 times higher) than that of 11181. When resources work on one trace at a time, the execution time and turnaround times will be almost equal. When a resource spends a significant amount of time doing multiple things, we see (1) *higher average execution times of activities* and (2) *larger differences between execution and turnaround times.* We notice this phenomenon for all resources who spend significant amounts of time working on multiple things simultaneously.

We have considered all resources who have spent at least 25 days on this process and analyzed their simultaneous working times and its influence on activity execution/turnaround times. There are 45 resources who have worked on this process 25 days or more. We divided the 45 resources into three classes based on their percentage of simultaneous working time. The top 15 resources who multitask frequently are labeled as *High* (and have an average percentage of simultaneous working time of 10.19%), the mid 15 resources as *Medium* (with an average percentage of simultaneous working time of 2.01%), and the bottom 15 resources as *Low* (with an average percentage of simultaneous working time of 0.27%). Table 6 depicts the average execution and turnaround times of resources

Table 5. Comparison of average execution and turnaround times between two resources: resource 11169 is frequently multitasking (21.55%) whereas resource 11181 is rarely working on multiple cases at the same time (2.62%).

	11169			11181		
Activity	No. exec.	Avg. exec. time (sec)	Avg. turnaround time (sec)	No. exec.	Avg. exec. time (sec)	Avg. turnaround time (sec)
W_Afhandelen leads	510	381	764	81	103	105
W_Beoordelen fraude	0	0	0	0	0	0
W_Completeren aanvraag	1069	613	927	1182	244	262
W_Nabellen incomplete dossiers	660	608	947	506	354	357
W_Nabellen offertes	217	430	573	1172	153	160
W_Valideren aanvraag	38	373	547	12	127	129

in these three classes for all the $W\_$ activities. From the table, we can clearly see a negative influence of multitasking on the execution/turnaround times, e.g., the average execution time of W Nabellen incomplete dossiers for resources who frequently multitask (class High) is 403 sec where as for resources who occasionally multitask (class Medium) it is 351 sec and it is 158 sec for resources who rarely multitask (class Low). Since $W\_$ activities are activities that correspond to contacting the customer, high turnaround times for these activities may result in unsatisfied customers.

Table 6. Comparison of average execution time (ET) and turnaround time (TAT) between resources with varying degrees of multitasking: the average percentage of simultaneous working time for the different classes are High (10.19%), Medium (2.01%), and Low (0.27%).

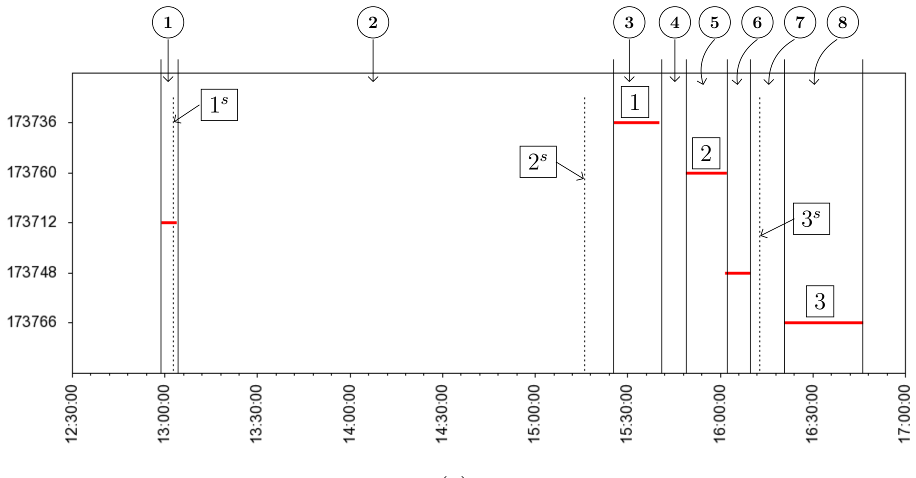
	High			Medium			Low		
Activity	No. exec.	Avg. ET (sec)	Avg. TAT (sec)	No. exec.	Avg. ET (sec)	Avg. TAT (sec)	No. exec.	Avg. ET (sec)	Avg. TAT (sec)
W_Afhandelen leads	2725	342	533	2001	151	165	225	237	240
W_Beoordelen fraude	0	0	0	0	0	0	247	66	67
W_Completeren aanvraag	9987	333	429	8889	299	315	516	443	454
W_Nabellen incomplete dossiers	3928	403	527	3919	351	366	2381	158	162
W_Nabellen offertes	7173	221	274	8093	173	184	3092	80	83
W_Valideren aanvraag	131	194	260	2784	930	952	4884	1034	1039

2.3 Resource–Idle Time Analysis

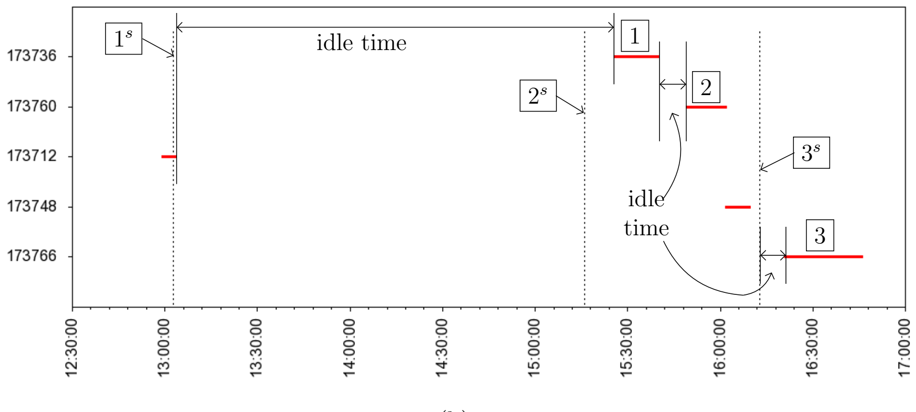
From Fig. 7, we can see that the resource is idle (i.e., not involved in executing any activity) in interval slots ②, ⑥, ⑧, and ⑭. It could be the case that these slots are free because no activity is available to be executed or alternatively, the resource is idle in spite of the availability of an activity to be executed. In this section, we analyze the time that resources are idle in spite of work available for execution. Enforcing the resources to utilize their idle times can help in reducing the cycle time of cases. To further analyze, resource idle times, we make the following assumptions: (i) *all resources that work on this process on a given day devote their full time to this process and not to any other process* and (ii) *the process is mostly sequential, i.e., an activity is assumed to be ready for execution soon after its predecessor has finished execution.*

For each resource, we consider its working slots on a given day. Let $w = [w_s, w_e]$ be a working slot (where w_s and w_e are the start and end times of the working slot) and $\hat{\mathbf{t}}$ be the corresponding activity instance, and let r be the resource executing $\hat{\mathbf{t}}$, i.e., activity instance $\hat{\mathbf{t}}$ is executed between w_s and w_e by resource r. Let $\hat{\mathbf{t}}^s$ be the time when the activity instance $\hat{\mathbf{t}}$ was ready to be executed (this is assumed to be the completion time of the activity preceding $\hat{\mathbf{t}}$ in the event log). Let $I_{\hat{\mathbf{t}}^s}$ be the set of interval slots that the resource is not involved in any activity since $\hat{\mathbf{t}}^s$ and before w_s. $I_{\hat{\mathbf{t}}^s}$ is the set of idle slots for the activity $\hat{\mathbf{t}}$. Ideally, the resource should have started executing $\hat{\mathbf{t}}$ in the first interval slot in $I_{\hat{\mathbf{t}}^s}$ after $\hat{\mathbf{t}}^s$. Fig. 10(a) depicts the working slots and interval slots for resource 11019 on 01-Oct-2011. The resource worked on five different traces on that day. Working slot $\boxed{1}$ corresponding to trace 173736 was executed between 15:25:56 and 15:40:09. However, the activity was scheduled to be executed at 13:02:44 ($\boxed{1^s}$ in the figure). Interval slot $\textcircled{2}$ corresponds to an idle slot for $\boxed{1^s}$. The resource was idle between 13:03:36–15:25:55 in spite of the fact that the activity $\boxed{1}$ was ready to be executed. Similarly, working slot $\boxed{2}$ corresponding to trace 173760 was scheduled at 15:15:57 but executed between 15:49:25 and 16:01:45. Now, there are two idle slots corresponding to interval slots $\textcircled{2}$ and $\textcircled{4}$ since $\boxed{2^s}$. The resource was idle between 15:15:58 and 15:25:55 in interval slot $\textcircled{2}$ and for the entire duration of interval slot $\textcircled{4}$, i.e., between 15:40:10 and 15:49:24 (i.e., the time between the completion of $\boxed{1}$ and the starting of $\boxed{2}$). Note that an interval slot can contribute to an idle slot of more than one activity. We resolve such ambiguities to the earliest working slot. In the above example, interval slot $\textcircled{2}$ is assigned to $\boxed{1}$ while only the interval slot $\textcircled{4}$ is considered to be an idle slot for working slot $\boxed{2}$. Working slot $\boxed{3}$ was ready for execution at 16:12:45 but actually executed between 16:21:09–16:45:56. The resource was not doing any activity between 16:09:36–16:21:08 though $\boxed{3}$ was ready to be executed from 16:12:46. Hence the resource is considered to be idle between 16:12:46 and 16:21:08. Fig. 10(b) depicts the idle times for the figure in (a).

We have analyzed the idle times for all resources in the event log. We considered only those activities that have both the start and complete event types (i.e., all W_ activities). We have ignored all idle slots that are one minute or less (we assume it is reasonable for an idle resource to initiate a task within one minute of its scheduling). Table 7 depicts the top five resources who have worked



(a)



(b)

Fig. 10. Working slots and corresponding activity's schedule time for resource 11019 on 01-Oct-2011.

on this process for at least 60 days and who were idle for a significant amount of time in spite of some activity being available for execution. The second column indicates the number of days that the resource worked on this process while the third column indicates the number of activities executed by the resource. The fourth column indicates the amount of time that a resource spent on this process working on some activity while the fifth column indicates the amount of time the resource was idle in spite of an activity being available for execution. The sixth column indicates the percentage of idle time with respect to the working time. The idle time also counts the time that a resource might have taken for breaks during a day (lunch/coffee break). To factor in this aspect, we discount 1 hr per work day of the resource. The adjusted idle time indicates the time that a resource was idle after factoring in break times (this is derived by subtracting the number of days (column 2) from the idle time (column 5)). Fig. 11 depicts the histogram of number of resources and the percentage of adjusted idle time spent by them. We can see that the majority of resources are idle for about 15 to 55 percent of their working time while some activity is available for execution. This impacts the cycle time of the cases. The organization should consider improving its process by reducing unjustified idle times.

Table 7. Top five resources based on the percentage of adjusted idle time.

Resource	No. Days	No. Activities	Working Time (hr)	Idle Time (hr)	Perc. Idle Time (%)	Adjusted Idle Time (hr)	Perc. Adj. Idle Time (%)
11259	81	1315	321.09	258.06	80.37	177.06	55.14
11049	74	1641	331.58	255.26	76.98	181.26	54.67
10899	79	1215	315.78	245.80	77.84	166.80	52.82
10982	99	1995	435.38	245.79	56.45	146.79	33.71
10913	106	2502	549.91	276.53	50.29	170.53	31.01

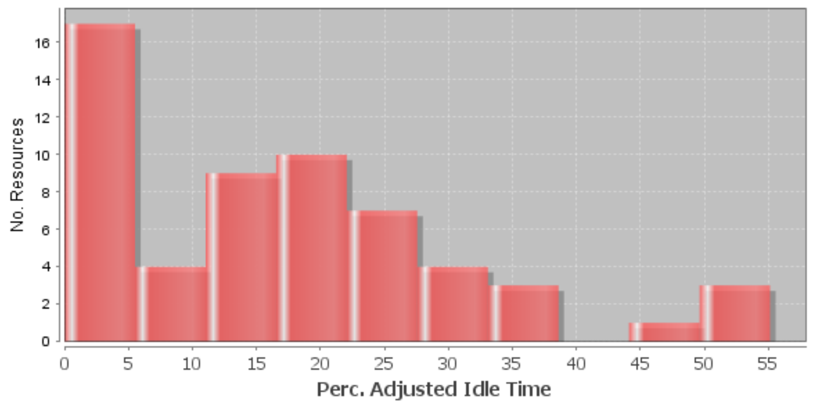


Fig. 11. Histogram of percentage of adjusted idle time spent by resources.

3 Process Diagnostics Using Trace Alignment

In Section 2, we focussed on resource behavior without considering the control-flow. Now, we shift our attention to the ordering of activities. In particular, we search for deviations and other non-conforming behavior. Trace alignment has been proposed as a powerful technique for process diagnostics [3, 4]. Trace alignment can be used to explore the process in the early stages of analysis. It can also be used to answer specific questions in later stages of analysis. For example, trace alignment can assist in answering diagnostic questions such as:

– *What is the most common (likely) process behavior that is executed?*
– *Are there any common patterns of execution in the traces?*
– *Where do process instances deviate and what do they have in common?*
– *What are typical contexts in which an activity or a set of activities tend to occur?*
– *What are the process instances that share/capture a desired behavior either exactly or approximately?*
– *Are there particular patterns (e.g., milestones, concurrent activities etc.) in the process?*

We have analyzed the financial event log using the Trace Alignment with Guide Tree plug-in in ProM [4]. We have analyzed the entire event log as well as subsets of homogenous cases based on the classification defined in Fig. 1. Analyzing homogenous subsets of cases instead of the entire event log has several advantages. Firstly, trace alignment is computationally expensive (polynomial in terms of the number of cases and the average length of a case) [4]; analyzing smaller homogenous subsets reduces the overall complexity. Secondly, heterogeneity in event logs can lead to disturbances in the alignment [4]; analyzing homogenous cases will lead to high-quality alignments. In the following sections, we discuss the results obtained using trace alignment on some categories of cases where we uncovered interesting insights on potential non-conforming behavior.

3.1 Cases Declined After Making an Offer

We have considered the event log containing cases that have been declined after making an offer and not suspected for fraud (cf. Fig. 1).[6] The selected event log contains 796 cases and $29,852$ events distributed over 27 activities. We have further partitioned the event log into six clusters (primarily for legibility purposes). Fig. 12 depicts a snippet of the alignment for one of the clusters (the alignment has been clipped for legibility reasons[7] and to show only the portion relevant for the discussion that follows). The alignment clearly uncovers common execution patterns present in the event log. In the remainder, we focus more on deviations in the event log. We can see that in only one trace (trace 204586), activity a (W_Completeren aanvraag-START) follows activity x (A_ACCEPTED-COMPLETE). In all other traces there is no a after x. It is surprising that the application is accepted before some requests for checking completeness are made with the customer. We used the LTL checker to check in how many of the traces do we see this exceptional behavior. We have considered both only the declined

[6] Recall that we classify a case into "Fraud" or "No Fraud" depending on the occurrence of task W_Beoordelen fraude.
[7] this cluster has 40 traces and the total alignment length is 114.

cases and the entire log for this analysis. Table 8 depicts the results of this analysis. We can see that only a negligible fraction of cases in the event log satisfy this relation suggesting that the position of `x` in trace 204586 is an outlier and a deviation from the normal behavior.

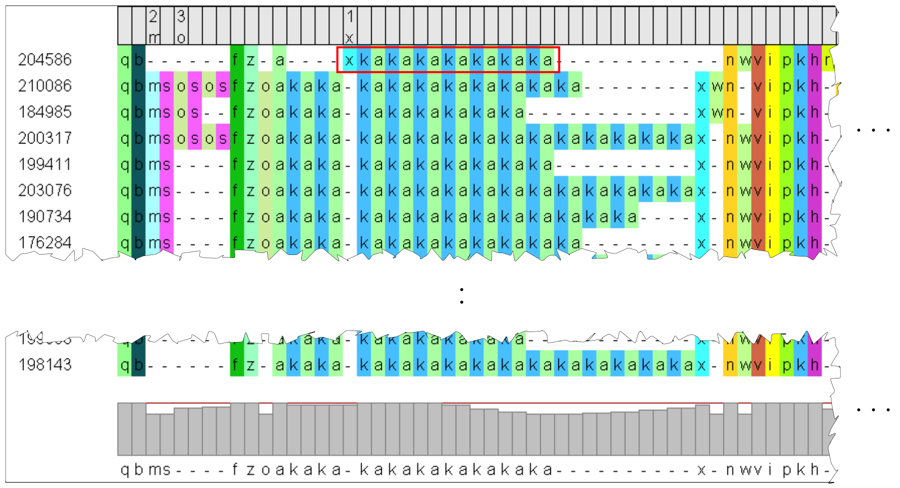

204586
210086
184985
200317
199411
203076
190734
176284
198143
q b m s - - - - f z o a k a k a - k a k a k a k a k a k a k a - - - - - - - - - - x - n w v i p k h -

Fig. 12. Snippet of the alignment of cases that have not been suspected for fraud and declined after making an offer. Also highlighted in the figure is an example of outlier behavior (see position of `x` in trace 204586).

Table 8. LTL constraint checking of W_Completeren aanvraag-START follows A_ACCEPTED-COMPLETE

Event Log	# Cases	No. Satisfied	Perct. Satisfied
Only offered and declined and no fraud	796	13	1.60%
All declined cases	7635	17	0.22%
Entire event log	13087	74	0.56%

3.2 Cases Cancelled Without Making an Offer

For our second analysis, we applied trace alignment to all cases that were cancelled without making an offer and not suspected for fraud (cf. Fig. 1). The selection corresponds to 1163 cases and 22132 events distributed over 11 activities. We have applied trace alignment on this selection and divided the log into five clusters. Fig. 13 depicts a snippet of the alignment for one of the clusters. This cluster contains 154 traces. Fig. 13 also highlights some of the

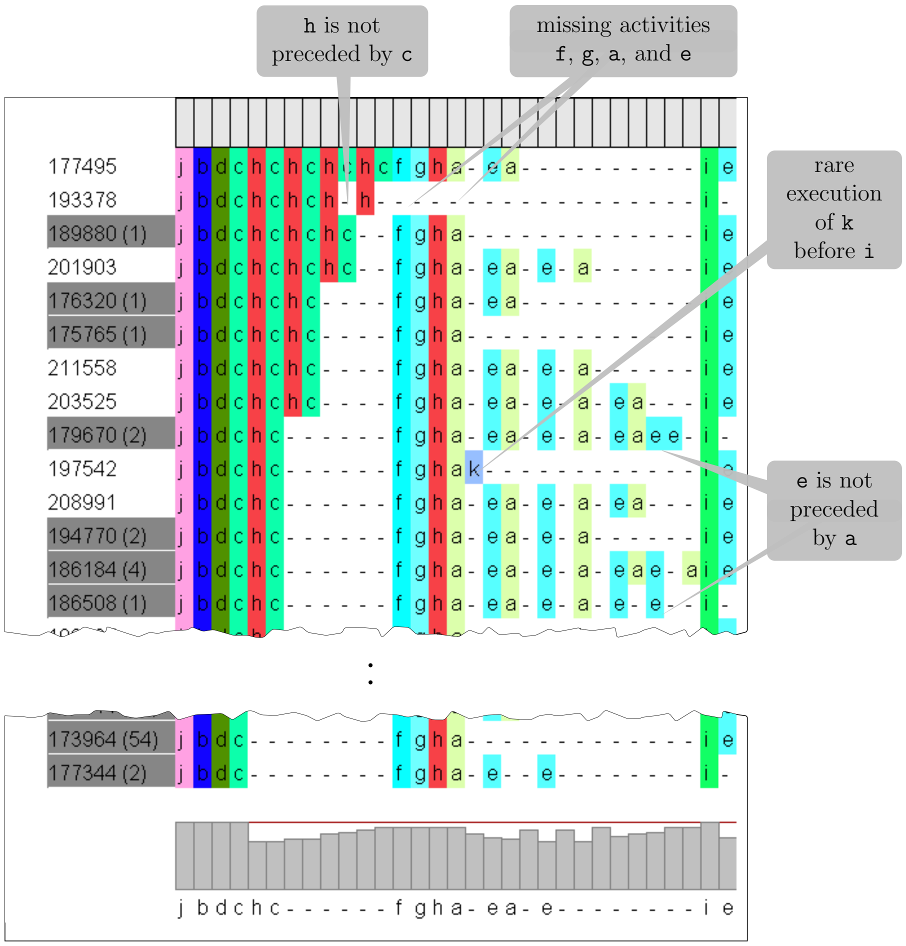


Fig. 13. Snippet of the alignment of cases that have been cancelled without making an offer. Also highlighted in the figure are outliers/exceptional behavior.

exceptional execution behavior. For example, we can see that in the second trace (trace name: 193378), the activities `f` (A_PREACCEPTED-COMPLETE), `g` (W_Completeren aanvraag-SCHEDULE), `a` (W_Completeren aanvraag-START), and `e` (W_Completeren aanvraag-COMPLETE) are missing before `i` (A_CANCELLED-COMPLETE), i.e., the case is cancelled without performing these activities. We have applied LTL checker to see how exceptional is this behavior in the entire log. Table 9 depicts the results of checking the properties (see the first two rows). We can see that only the trace 193378 captured in this alignment violates the normative behavior, clearly indicating that this trace is an outlier. Similarly, in the same trace, there is a missing activity `c` (W_Afhandelen leads-START) before the activity `h` (W_Afhandelen leads-COMPLETE). In the entire event log only one trace exhibits this behavior thereby indicating that this is most likely an outlier than an issue with logging.

We can further see that there is a rare execution of activity `k` (A_ACCEPTED) before `i` (A_CANCELLED) in trace 197542, i.e., an application is cancelled after it has been accepted. Table 9 depicts the number of such cases in the event log. We can see that in only a small fraction of cases (5.59%) an application is accepted before it is cancelled without making an offer. However, this is always the case in traces that have been cancelled after an offer is made. Another exception that we see is the absence of activity `a` (W_Completeren aanvraag-START) before `e` (W_Completeren aanvraag-COMPLETE). This might most likely be an issue with logging.

Table 9. Results of checking some LTL properties on cases that were cancelled

LTL Formula	Event Log	# Cases	No. Satisfied	Perct. Satisfied
A_Preaccepted does not Precede A_Cancelled	Cancelled without offer and no fraud	1163	1	0.08%
	All Cancelled cases	2807	1	0.03%
W_Completeren aanvraag does not Precede A_Cancelled	Cancelled without offer and no fraud	1163	1	0.08%
	All Cancelled cases	2807	1	0.03%
A_Accepted Precedes A_Cancelled	Cancelled without offer and no fraud	1163	65	5.59%
	Cancelled after an offer is made	1640	1640	100.00%
	All Cancelled cases	2807	1706	60.77%

3.3 Cases Cancelled After Making an Offer

We considered the cases that have been cancelled after an offer was made. The selected event log contains 1640 cases and 56,803 events distributed over 26 activities. Fig. 14 depicts a snippet of the aligned traces in one of the clusters of this event log (the event log is partitioned into 10 clusters). The cluster has 48 traces. From the alignment we can see that for one of the traces (trace name: 181607), we see the activity pairs `g` (W_Nabellen offertes-START) and `l` (W_Nabellen offertes-COMPLETE) following `z` (A_CANCELLED-COMPLETE), i.e., phone calls pertaining to offers were made to the customer in spite of the application having been cancelled. We have applied LTL checker to check how frequent is this behavior. Table 10 depicts the result of checking this constraint. We can see that this is a clear outlier with only two traces (trace names: 181607 and 201427) exhibiting this behavior. In this alignment, we also see that there is a missing `g` before an `l` in some traces, i.e., in some instances, a (W_Nabellen offertes-COMPLETE) activity is not preceded by its corresponding (W_Nabellen offertes-START) activity. This might most likely be an issue with logging.

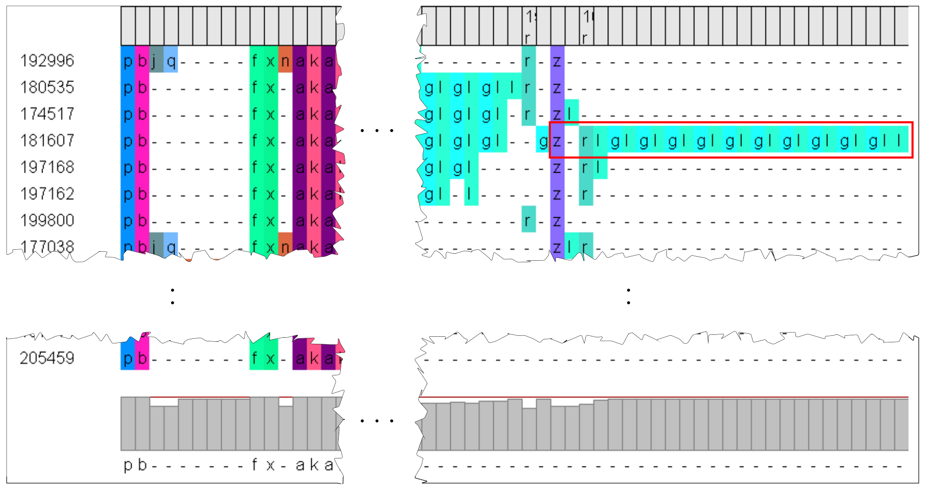


Fig. 14. Snippet of the alignment of cases that have been cancelled after an offer was made. Also highlighted in the figure is an outlier.

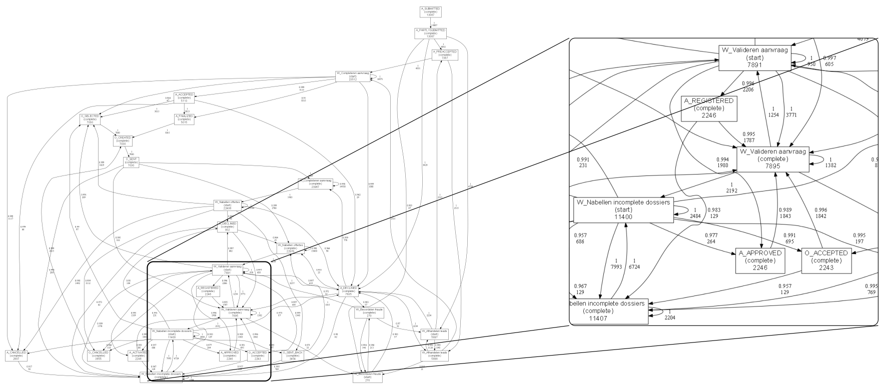
Table 10. LTL constraint checking of W_Nabellen offertes-START follows A_CANCELLED

Event Log	# Cases	No. Satisfied	Perct. Satisfied
Only cancelled after an offer cases	1640	2	0.12%
All cancelled cases	2407	2	0.08%

In this fashion, trace alignment can assist in uncovering extremely interesting insights and act as probes when analyzing process execution behavior thereby giving cues on process improvement opportunities.

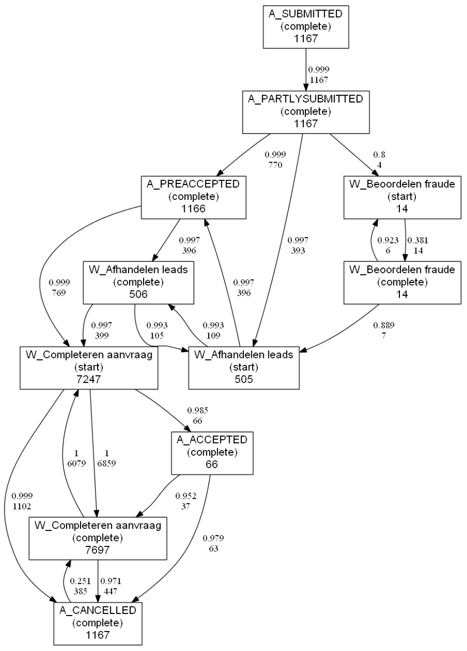
4 Control-Flow Analysis

In this section, we analyze the control-flow aspects of the given event log. When we mine for a process model using the event log as given, we get a spaghetti-like and completely incomprehensible model. Fig. 15 depicts the Heuristic net [5] mined from the event log after removing all schedule events (this filtered event log contains 13087 traces and 235882 events distributed over 29 activities). There are two primary reasons for such a spaghetti-like model: (i) *the event log is heterogenous* and (ii) *there is a lot of concurrency in the process*. Process mining results can be improved by partitioning the event log into homogeneous subsets of cases [6–10]. We use the classification defined in Section 1 and consider clusters of homogenous cases for control-flow analysis.



W_Valideren aanvraag (start) 7891
A_REGISTERED (complete) 2246
W_Valideren aanvraag (complete) 7895
W_Nabellen incomplete dossiers (start) 11400
A_APPROVED (complete) 2246
O_ACCEPTED (complete) 2243
bellen incomplete dossiers (complete) 11407

Fig. 15. Heuristic net mined on the whole log (after filtering schedule events).



A_SUBMITTED (complete) 1167
0.999 1167
A_PARTLYSUBMITTED (complete) 1167
0.999 770
0.8 4
A_PREACCEPTED (complete) 1166
W_Beoordelen fraude (start) 14
0.997 396
0.997 393
0.923 6
0.381 14
W_Afhandelen leads (complete) 506
W_Beoordelen fraude (complete) 14
0.999 769
0.997 396
0.997 399
0.993 105
0.993 109
0.889 7
W_Completeren aanvraag (start) 7247
W_Afhandelen leads (start) 505
0.985 66
A_ACCEPTED (complete) 66
1 6079
1 6859
0.999 1102
0.952 37
W_Completeren aanvraag (complete) 7697
0.979 63
0.251 385
0.971 447
A_CANCELLED (complete) 1167

Fig. 16. Heuristic net mined for cases that were cancelled without making an offer.

4.1 Traditional Process Discovery

Fig. 16 depicts the Heuristic net [5] mined using the cases that were cancelled without making an offer. This event log contains 1167 cases and 20716 events distributed over 11 activities (after removing the schedule events). This model is straightforward and comprehensible. An application is first pre-accepted and in some cases W_Afhandelen leads activity is executed. After an application is pre-accepted, W_completeren aanvraag activity is performed one or more times. An application is cancelled during some instance of execution of W_completeren aanvraag. Sometimes (only in 66 out of 1167 cases), an application is accepted before it is cancelled. This behavior is also uncovered using trace alignment and we suspect that this is most likely to be outlier behavior and probably non-conforming. In addition, for cases that are suspected for fraud, execution of the activity related to fraudulent checks, viz., W_Beoordelen fraude is performed.

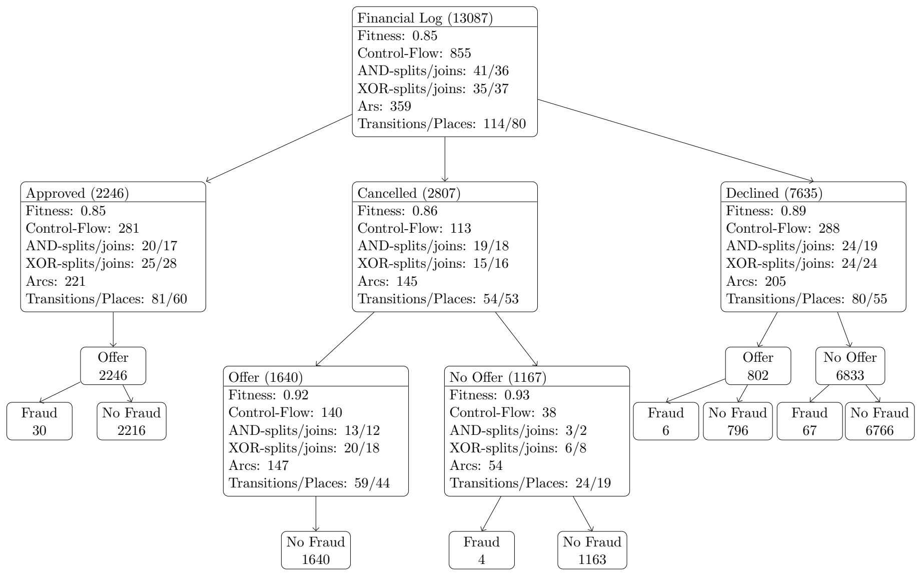


Fig. 17. Quality metrics for the various categories of cases. The fitness metric is obtained using the Conformance Checker plug-in while the structural complexity metrics are obtained using the Petri net Complexity Analysis plug-in in ProM 5.2. For the fitness metric, the higher the value, the better is the model. For the complexity metrics, the lower the values the better is the model.

We have applied traditional process discovery algorithms such as the Heuristics miner [5] and Petri net discovery algorithms [11] for the different classes of cases (depicted in Fig. 1) and in all the cases, we obtained more comprehensible and structurally simpler models when compared to analyzing the whole event log. Fig. 17 depicts the case classification tree along with some metrics assessing the quality of the mined models (we depict the metrics only for some subsets of homogenous cases due to space constraints). We have used the Heuristics miner [5] for all the scenarios and translated the Heuristic net into Petri net and applied the Conformance Checker plug-in [12] in ProM 5.2 for the fitness metric. The complexity metrics are obtained using the Petri net Complexity Analysis plug-in in ProM 5.2. From the figure, we can see that as we go down the tree (i.e., as we reduce the heterogeneity), we get models that are more fitting and structurally simpler.[8]

4.2 Discovering Process Maps

The discovered process models can further be simplified by using the two-phase approach to process discovery [13, 14]. In the first phase, we define abstractions of activities based on common execution patterns manifested in the event log. The event log is transformed using these abstractions wherein patterns that define abstractions are replaced by its corresponding abstract activity. At the same time, a sub-log is created for each abstract activity wherein the process instances are defined by the pattern manifestations that are replaced. In the second phase a process map is mined over the abstracted log. We use the Pattern Abstractions and Fuzzy Map Miner plug-ins in ProM 6.0 for the two-phase approach to process discovery.

We consider the event log pertaining to cases that were approved. The event log contains 2246 cases and 88570 events distributed over 26 activities (after removing the schedule events). We have discovered the tandem arrays [15] in the log and defined abstractions over them. Six abstractions pertaining to the start and complete event types of different W_ activities are formed. The event log is transformed using these abstractions [13]. The transformed log contains 2246 cases and 44205 events distributed over 20 activities. Fig. 18 depicts the process map obtained on the transformed log. Blue (dark) colored nodes in the map are abstract activities which can be zoomed in to see the sub-processes underneath them. In this example, the sub-processes are simple loops (since we formed abstractions over tandem arrays) between the start and complete event types of the corresponding W_ activities. The process is easy to comprehend. An application that is submitted is first pre-accepted. If needed, the checks W_Afhandelen Leads and W_Completeren aanvraag are performed. The application is then finalized and accepted and an initial offer is made and sent to the customer. Offers can be cancelled and created multiple times; the activity W_Nabellen offertes is involved in this process. The application is subjected to further checks one/more times W_Valideren aanvraag and W_Nabellen incomplete dossiers before it is activated, approved, and registered. Fig. 19 and Fig. 20 depicts the process maps for cases that were cancelled and declined respectively.

As shown, we can deal with heterogeneity by partitioning the event log into homogenous subsets of cases. Analysis of each subset separately enables the discovery of comprehensible process models. Even simpler models can be obtained by using the two-phase approach to process discovery, which enables the discovery of hierarchical process models.

[8] several of the transitions in the Petri net obtained from Heuristic net are silent transitions.

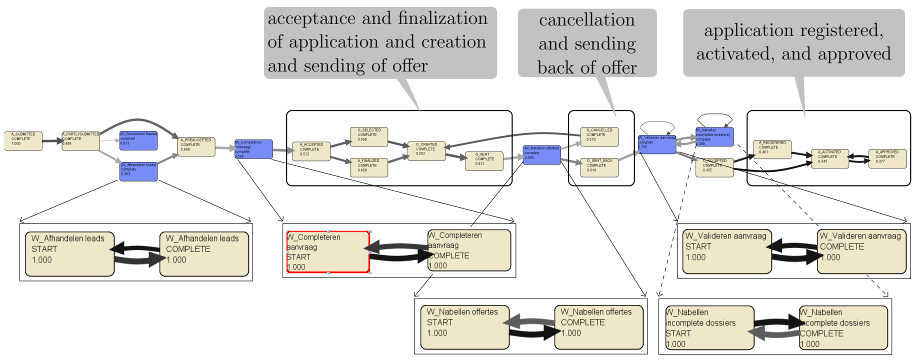

acceptance and finalization of application and creation and sending of offer
cancellation and sending back of offer
application registered, activated, and approved
W_Afhandelen leads START 1.000
W_Afhandelen leads COMPLETE 1.000
W_Completeren aanvraag START 1.000
W_Completeren aanvraag COMPLETE 1.000
W_Valideren aanvraag START 1.000
W_Valideren aanvraag COMPLETE 1.000
W_Nabellen offertes START 1.000
W_Nabellen offertes COMPLETE 1.000
W_Nabellen incomplete dossiers START 1.000
W_Nabellen incomplete dossiers COMPLETE 1.000

Fig. 18. Process map of cases that were approved.

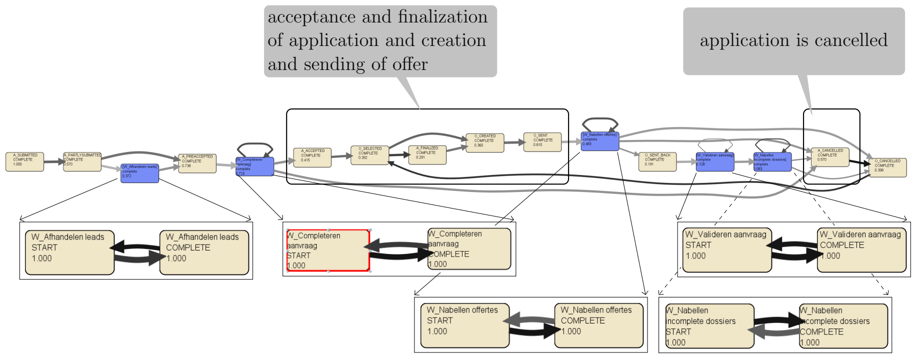

acceptance and finalization of application and creation and sending of offer
application is cancelled
W_Afhandelen leads START 1.000
W_Afhandelen leads COMPLETE 1.000
W_Completeren aanvraag START 1.000
W_Completeren aanvraag COMPLETE 1.000
W_Valideren aanvraag START 1.000
W_Valideren aanvraag COMPLETE 1.000
W_Nabellen offertes START 1.000
W_Nabellen offertes COMPLETE 1.000
W_Nabellen incomplete dossiers START 1.000
W_Nabellen incomplete dossiers COMPLETE 1.000

Fig. 19. Process map of cases that were cancelled.

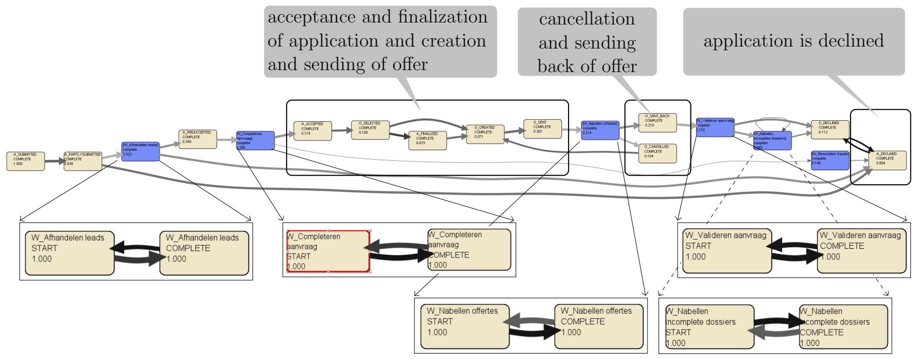

acceptance and finalization of application and creation and sending of offer
cancellation and sending back of offer
application is declined
W_Afhandelen leads START 1.000
W_Afhandelen leads COMPLETE 1.000
W_Completeren aanvraag START 1.000
W_Completeren aanvraag COMPLETE 1.000
W_Valideren aanvraag START 1.000
W_Valideren aanvraag COMPLETE 1.000
W_Nabellen offertes START 1.000
W_Nabellen offertes COMPLETE 1.000
W_Nabellen incomplete dossiers START 1.000
W_Nabellen incomplete dossiers COMPLETE 1.000

Fig. 20. Process map of cases that were declined.

5 Implementation and Tools for Analysis

For our analysis, we used three types of ProM plug-ins:

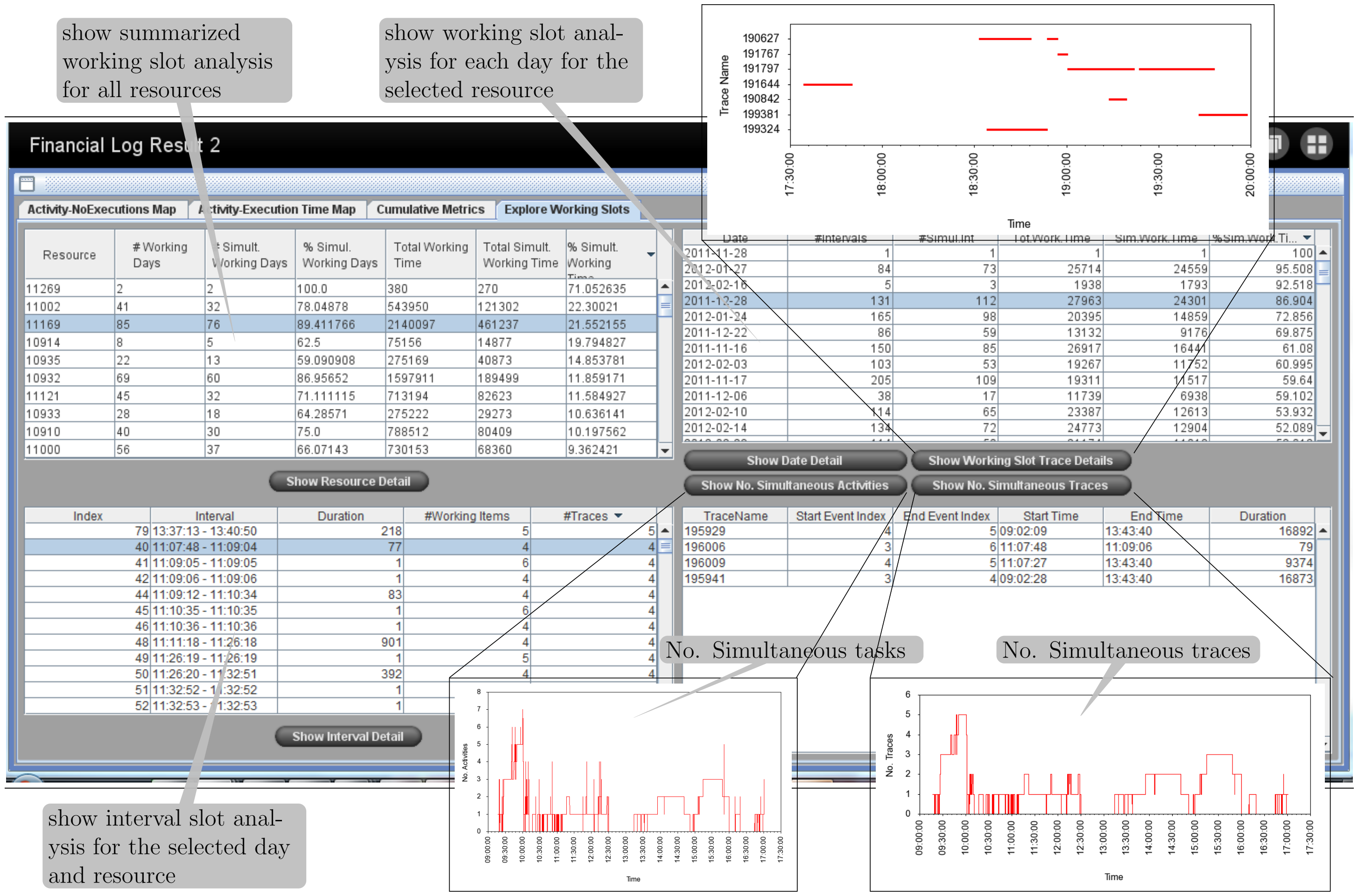
– *A dedicated plug-in developed for this challenge:* We have implemented the ideas presented in this paper on the analysis of resources (Sections 1 and 2) as the Resource Work Analysis plug-in[9] in ProM 6.0. Fig. 21 depicts the result for working slot analysis from the plug-in.
– *Recently developed plug-ins by authors:* We have used several plug-ins developed by us during our earlier research for the analysis of the BPI Challenge event log. The Guide Tree Miner and the Trace Alignment with Guide Tree plug-ins are used for aligning the traces and exploring the alignment for deviations and non-conforming behavior. Trace alignment is a two step process: first, we group similar traces in clusters [6, 7]; second, we visualize these clusters by aligning the traces [3, 4]. The Guide Tree Miner plug-in clusters the traces based on the techniques proposed in [6, 7, 13] while the Trace Alignment with Guide Tree plug-in uses this guide tree and aligns the traces.

  For the discovery of process maps, we use the Pattern Abstractions and Fuzzy Map Miner plug-ins. Together, these two plug-ins realize the two-phase approach to process discovery [14] and enables the discovery of hierarchical process models [16, 17]. The Pattern Abstractions plug-in caters to the discovery of common execution patterns, the definition of abstractions over them, and the transformation of the event log with these abstractions. During the transformation phase, the Pattern Abstractions plug-in generates a sub-log that captures the manifestation of execution patterns defined by that abstraction as its process instances. The Fuzzy Map Miner plug-in is an enhancement of the Fuzzy miner [18] and utilizes the availability of sub-logs to mine sub-processes, which are displayed upon zooming-in on abstract activities.
– *General purpose plug-ins:* We have also used several general purpose ProM plug-ins for process discovery and conformance checking. More specifically, we have used the Heuristics Miner [5], Conformance Checker [12], Petri net Complexity Analysis, and LTL Checker plug-ins in ProM 5.2

6 Conclusions

In this paper, we used a systematic approach to analyze the financial institute event log provided for the BPI challenge. We proposed a hierarchical classification of the event log based on the loan/overdraft application characteristics and showed that this classification simplifies the complexity of analysis and enables the discovery of interesting insights. We have reported the results (and observations) from the analysis of the log on three different aspects (a) resource (b)

[9] The plug-in can be provided upon request.


show summarized working slot analysis for all resources
show working slot analysis for each day for the selected resource
show interval slot analysis for the selected day and resource
No. Simultaneous tasks
No. Simultaneous traces
Financial Log Result 2
Activity-NoExecutions Map
Activity-Execution Time Map
Cumulative Metrics
Explore Working Slots
Show Resource Detail
Show Interval Detail
Show Date Detail
Show Working Slot Trace Details
Show No. Simultaneous Activities
Show No. Simultaneous Traces
Trace Name
Time
No. Activities
No. Traces

Resource	# Working Days	# Simult. Working Days	% Simul. Working Days	Total Working Time	Total Simult. Working Time	% Simult. Working Time
11269	2	2	100.0	380	270	71.052635
11002	41	32	78.04878	543950	121302	22.30021
11169	85	76	89.411766	2140097	461237	21.552155
10914	8	5	62.5	75156	14877	19.794827
10935	22	13	59.090908	275169	40873	14.853781
10932	69	60	86.95652	1597911	189499	11.859171
11121	45	32	71.111115	713194	82623	11.584927
10933	28	18	64.28571	275222	29273	10.636141
10910	40	30	75.0	788512	80409	10.197562
11000	56	37	66.07143	730153	68360	9.362421

Date	#Intervals	#Simul.Int	Tot.Work.Time	Sim.Work.Time	%Sim.Work.Ti...
2011-11-28	1	1	1	1	100
2012-01-27	84	73	25714	24559	95.508
2012-02-16	5	3	1938	1793	92.518
2011-12-28	131	112	27963	24301	86.904
2012-01-24	165	98	20395	14859	72.856
2011-12-22	86	59	13132	9176	69.875
2011-11-16	150	85	26917	16441	61.08
2012-02-03	103	53	19267	11752	60.995
2011-11-17	205	109	19311	11517	59.64
2011-12-06	38	17	11739	6938	59.102
2012-02-10	114	65	23387	12613	53.932
2012-02-14	134	72	24773	12904	52.089

Index	Interval	Duration	#Working Items	#Traces
79	13:37:13 - 13:40:50	218	5	5
40	11:07:48 - 11:09:04	77	4	4
41	11:09:05 - 11:09:05	1	6	4
42	11:09:06 - 11:09:06	1	4	4
44	11:09:12 - 11:10:34	83	4	4
45	11:10:35 - 11:10:35	1	6	4
46	11:10:36 - 11:10:36	1	4	4
48	11:11:18 - 11:26:18	901	4	4
49	11:26:19 - 11:26:19	1	5	4
50	11:26:20 - 11:32:51	392	4	4
51	11:32:52 - 11:32:52	1		
52	11:32:53 - 11:32:53	1		

TraceName	Start Event Index	End Event Index	Start Time	End Time	Duration
195929	4	5	09:02:09	13:43:40	16892
196006	3	6	11:07:48	11:09:06	79
196009	4	5	11:07:27	13:43:40	9374
195941	3	4	09:02:28	13:43:40	16873

Fig. 21. Screenshot of the plug-in's result for working slot analysis.

control-flow and (c) process diagnostics. Only a small subset of our findings are reported in this paper. However, we strongly believe that more accurate insights and interpretations can be obtained/inferred by performing the analysis in collaboration with the domain experts.

Key Findings

- Some resources tend to work on cases (loan/overdraft applications) simultaneously and spend significant amounts of their working time executing multiple activities. The top five resources exhibiting such multitasking behavior are 11002, 11169, 10932, 11121, and 10910.
- Working on simultaneous cases has a negative influence on the execution times of activities leading to high turnaround times. Obviously, this is undesirable for customers and the organization.
- The average execution and turnaround times of activities is much higher for resources who work on simultaneous cases when compared to their counterparts who work on one case at a time for similar workloads.
- Several resources are often idle although an activity is available for execution. This impacts the cycle time of cases. The top five resources who are often idle while work is piling up are 11259, 11049, 10899, 10982, and 10913.
- At first glance, the event log may seem complex due to the heterogeneity in the log. However, our hierarchical classification of the log based on the characteristics of the loan/overdraft applications helps to simplify analysis significantly.
- Analyzing homogenous subsets of cases in the event log based on the classification proposed in this paper reveals that the process is in fact rather simple. Comprehensible process models and interesting diagnostic insights can be uncovered using such a classification.
- There are several (potential) outliers in the event log
  - an automated resource (112) is involved in the *approval* of 3 loan applications.
  - in two cases, W_Nabellen offertes is executed even after the application is cancelled.
  - in one case, an application is cancelled even before it is pre-accepted.
  - in one case, completion checks (W_Completeren aanvraag) are performed even after the application is cancelled (alternatively, an application is cancelled without thorough checks).
  - in 74 cases in the entire event log, completion checks (W_Completeren aanvraag) are performed even after the application is accepted (alternatively, applications are accepted without thorough checks).
- There are some anomalies with respect to the quality of the log (missing information, ambiguities, etc.). The organization should take appropriate steps to improve this.

References

1. Nakatumba, J., van der Aalst, W.M.P.: Analyzing Resource Behavior Using Process Mining. In Rinderle-Ma, S., Sadiq, S.W., Leymann, F., eds.: Business Process Management Workshops. Volume 43 of Lecture Notes in Business Information Processing., Springer-Verlag, Berlin (2010) 69–80
2. Nakatumba, J., Westergaard, M., van der Aalst, W.M.P.: Generating Event Logs with Workload-Dependent Speeds from Simulation Models. In Bajec, M., Eder, J., eds.: CAiSE Workshops. Volume 112 of Lecture Notes in Business Information Processing., Springer-Verlag, Berlin (2012) 383–397
3. Bose, R.P.J.C., van der Aalst, W.M.P.: Trace Alignment in Process Mining: Opportunities for Process Diagnostics. In Hull, R., Mendling, J., Tai, S., eds.: Proceedings of the 8th International Conference on Business Process Management (BPM). Volume 6336 of LNCS., Springer-Verlag (2010) 227–242
4. Bose, R.P.J.C., van der Aalst, W.M.P.: Process Diagnostics Using Trace Alignment: Opportunities, Issues, and Challenges. Information Systems **37**(2) (2012) 117–141
5. Weijters, A.J.M.M., van der Aalst, W.M.P.: Rediscovering Workflow Models From Event-based Data Using Little Thumb. Integrated Computer-Aided Engineering **10**(2) (2003) 151–162
6. Bose, R.P.J.C., van der Aalst, W.M.P.: Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models. In: Business Process Management Workshops. Volume 43 of LNBIP., Springer (2010) 170–181
7. Bose, R.P.J.C., van der Aalst, W.M.P.: Context Aware Trace Clustering: Towards Improving Process Mining Results. In: Proceedings of the SIAM International Conference on Data Mining (SDM). (2009) 401–412
8. de Medeiros, A.K.A., Guzzo, A., Greco, G., van der Aalst, W.M.P., Weijters, A.J.M.M., van Dongen, B.F., Sacca, D.: Process Mining Based on Clustering: A Quest for Precision. In ter Hofstede, A.H.M., Benatallah, B., Paik, H., eds.: Business Process Management Workshops. Volume 4928 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2008) 17–29
9. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Discovering Expressive Process Models by Clustering Log Traces. IEEE Transactions on Knowledge and Data Engineering **18**(8) (2006) 1010–1027
10. Weerdt, J.D., vanden Broucke, S.K.L.M., Vanthienen, J., Baesens, B.: Leveraging Process Discovery with Trace Clustering and Text Mining for Intelligent Analysis of Incident Management Processes. In: 2012 IEEE Congress on Evolutionary Computation (CEC). (2012) 1–8
11. van Dongen, B.F., de Medeiros, A.K.A., Wen, L.: Process Mining: Overview and Outlook of Petri Net Discovery Algorithms. In Jensen, K., van der Aalst, W.M.P., eds.: T. Petri Nets and Other Models of Concurrency. Volume 5460 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2009) 225–242
12. Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems **33**(1) (2008) 64–95
13. Bose, R.P.J.C.: Process Mining in the Large: Preprocessing, Discovery, and Diagnostics. PhD thesis, Eindhoven University of Technology (2012)
14. Li, J., Bose, R.P.J.C., van der Aalst, W.M.P.: Mining Context-Dependent and Interactive Business Process Maps using Execution Patterns. In zur Muehlen, M., Su, J., eds.: BPM 2010 Workshops. Volume 66 of LNBIP., Springer-Verlag (2011) 109–121

15. Bose, R.P.J.C., van der Aalst, W.M.P.: Abstractions in Process Mining: A Taxonomy of Patterns. In Dayal, U., Eder, J., Koehler, J., Reijers, H., eds.: Business Process Management. Volume 5701 of LNCS., Springer-Verlag (2009) 159–175
16. Bose, R.P.J.C., Verbeek, E.H.M.W., van der Aalst, W.M.P.: Discovering Hierarchical Process Models Using ProM. In Nurcan, S., ed.: CAiSE Forum 2011. Volume 107 of Lecture Notes in Business Information Processing., Springer-Verlag, Berlin (2012) 33–48
17. Bose, R.P.J.C., Verbeek, E.H.M.W., van der Aalst, W.M.P.: Discovering Hierarchical Process Models Using ProM. In Nurcan, S., ed.: Proceedings of the CAiSE Forum. Volume 734., CEUR-WS.org (2011) 33–40
18. Günther, C.W., van der Aalst, W.M.P.: Fuzzy Mining: Adaptive Process Simplification Based on Multi-perspective Metrics. In Alonso, G., Dadam, P., Rosemann, M., eds.: Business Process Management. Volume 4714 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2007) 328–343