Process Mining in IT Service Management: A Case Study

Borja Vázquez-Barreiros[1], David Chapela[1], Manuel Mucientes[1], Manuel Lama[1], and Diego Berea[2]

[1] Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS) Universidade de Santiago de Compostela. Santiago de Compostela, Spain
{borja.vazquez, david.chapela, manuel.mucientes, manuel.lama}@usc.es
[2] Ozona Consulting. Santiago de Compostela, Spain
diego.berea@ozonaconsulting.com

Abstract. The explosion of process-related data in nowadays organizations has raised the interest to exploit the data in order to know in deep how the business processes are being carried out. To face this need, process mining has emerged as a way to analyze the behavior of an organization by extracting knowledge from process related data. In this paper, we present a case study of process mining in a real IT service management scenario. We describe an exploratory analysis of real life event logs generated between 2012 and 2015, in four different processes designed within an IT platform. More specifically, we analyze the way of handling the different requests and incidents registered in an organization.

1 Introduction

In the recent years, there has been a huge investment in developing technologies to automate the different tasks carried out in an organization, and to store all the possible information generated during these tasks. In particular, regarding business processes, this has lead to an incredible growth on the amount of process-related data, i.e., execution traces of business activities. Clearly, the explosion of this kind of information has opened a door to provide insights into the actual way of working in an organization, to predict performance using simulation, to detect deviations in the process or to improve the way certain business activities are executed [1].

Within this context, a business process, henceforth a process model, is understood as a collection of related structured activities that produce a specific outcome, e.g., a product or a service. Typically, process models have a detailed description prescribing how tasks must or should be done, i.e., they describe the way of working. Unfortunately, there might be differences between the designed process model, and how the process is being executed in reality [1]. Hence, creating process models is a difficult and error-prone task, that can lead to an evident gap between *what we think is going on* (the a-priori process model) and *what is really happening* (the real process model). With this in mind, process mining has emerged as a way to analyze the behavior of an organization by extracting knowledge from process-related data, and offering techniques to discover, monitor and enhance real processes [1]. Nowadays we can find several academic (PMLAB[3]), open-source (ProM[4]), and commercial (Disco[5]) tools featuring an extensive set of analysis techniques for process mining.

Concerning its real *applicability*, process mining has been widely applied in multiple domains, showing an incredible potential as a link between Business Process Management and all kinds of analytical techniques that are not necessarily process-aware. Examples of this success can be found in a wide variety of fields[6]. In the literature, we can find several case studies providing insights into hospital and health care processes [2,3,4,5]; education [6,7]; manufacturing processes [8]; invoice verification processes in SAP [9]; in monitoring sea regulations [10]; financial services [11] or purchases process in IBM [12].

Regarding this paper, we present an experience of applying process mining in a real IT service management (ITSM) scenario. The dataset used in this case study consists of a total of 2,004 different requests and incidents generated in an organization between 2012 and 2015, and four different process models designed to handled such requests and incidents recorded in the system. Within this scenario, we will follow a *data-driven* approach, seeking new insights and generating ideas and hypotheses for further research, i.e., this analysis has an exploratory character. Thus, the main idea behind this study is to find the gap between the modeled behavior, i.e., the *as-designed* process model or what the organization thinks is happening; and the observed behavior, i.e., the *as-is* process model or what is really happening. In this analysis, we mainly use the academic tool ProDiGen [13] platform[7] and the open source tool ProM.

2 Case study

The *case under study* presented in this paper is the analysis of the services to handle requests and incidents in a real organization. In particular, this organization has implemented a central point of contact for handling customers, users and other issues within the company, e.g., stock capacity, changes on a particular process, financial management, etc. In order to plan, deliver, operate and control the offered IT services, this organization relies on a ITSM software platform. This platform provides a wide range of management services. In particular, it implements the means for monitoring the progress of different events, i.e., it logs process-related data. However, the ITSM tool does not provide any kind of process mining analysis technique, that is, it does not fully exploit this kind of

[3] `https://www.cs.upc.edu/~jcarmona/PMLAB/`

[4] `http://www.promtools.org`

[5] `https://fluxicon.com/disco`

[6] The IEEE CIS Task Force on Process Mining has an extensive compilation of different successful case studies `http://www.win.tue.nl/ieeetfpm/doku.php?id=shared:process_mining_case_studies`.

[7] `https://tec.citius.usc.es/processmining`

information. Hence, although plenty of process-related information is available, this organization has no clear idea of how the incidents and requests are handled.

In this system, requests and incidents, henceforth *tickets*, are usually mapped to one or more management processes, i.e., process models. Thus, when a ticket is registered in the system, the linked management process indicates the *actions* to be performed. Each one of these process models is generated through a graphical tool, in which the designer defines, configures and organizes the different steps in such a process model. Hence, each of these management processes specifies a guideline tof steps indicating who, how and when should intervene during the processing of a ticket. In other words, in order to be properly handled, each ticket has to perform a sequence of defined steps. From the point of view of process mining, each one of this steps can be considered as an *event*, i.e., a specific activity in the system, and each ticket as a *trace*, i.e., a sequence of events. Therefore, for each defined process model in this organization, it is possible to extract an *event log*, i.e., a group of traces that consist of process events.

Concerning this case study, we identify four major process models: *Issues resolution*, *Orders resolution*, *Standard changes* and *Emergency changes*. Henceforth we denote this processes as: *Workflow 1*, *Workflow 2*, *Workflow 3*, *Workflow 4*, respectively. In total, this four processes are mapped to 2,004 tickets, i.e., traces, registered between 2012 and 2015. Hence, we are dealing with historic data, i.e., traces that, in theory, have been completed. Within this scenario, the main motivation is to obtain, from an exploratory point of view, insights into how the different tickets are being handled in reality, and if they conform with what was designed. In other words, we aim to *check* and *compare* reality with what was planned. Remark that this exploratory analysis does not have a specific goal, i.e., through this analysis we do not aim to intervene, adjust or redesign the possible deviations and differences between modeled behavior and reality, but rather gain insights for a further *question-driven* and deeper analysis.

We have executed the following steps to perform this exploratory case study. First, we extracted and filtered the data from the ITSM tool database. Then, we conformed reality with the a-priori process models. After this, we applied discovery techniques to retrieve the real processes. Finally, we measured the performance of the processes to detect bottlenecks and performance issues.

2.1 Data preparation

The first phase in the presented analysis consists of the preparation and exploration of the available process data between 2012 and 2015. Hence, in this step, the data was extracted from the database of the platform, and converted into a standard event log storage format, i.e., XES [14]. Furthermore, we also translated the a-priori process models to their equivalent Petri net representation.

During this preprocessing step, we detected several traces that, during their lifespan, were linked to different process models. In other words, a ticket was initially handled through the steps of a specific process model and, at some point, was transferred to a different process model, having to start again. The main problem behind this particular behavior is that there is no clear indicator

Table 1: Event logs characteristics.

	Workflow 1	***Workflow 2***	***Workflow 3***	***Workflow 4***
#activities	5	7	12	7
#traces	158	1,151	84	611
#events	886	7,242	696	3,580

#activities, #traces and #events stands for the number of activities, process instances, and events, respectively, in each event log.

in the database on when a change of this type took place: the only information is when a ticket fired an step on a different process model. Clearly, a deeper study would involve analysing these particular cases and checking, for instance, the impact on the average time resolution of the tickets, or the reason behind these transfers. However, in this case study, we filtered this kind of behavior, focusing the analysis on those tickets handled only through a single process model.

Once correctly identified all the different tickets, i.e., the traces, we found a problem related with the ordering of the events. Specifically, some events in the system lacked the *start timestamp* attribute, precluding the creation of the actual trace of events for each ticket. Note that we are working with *atomic* activities, and we sort the events of a trace based on the *start timestamp*. In particular, we detected that this behavior was always related to automatic activities. Hence, in order to properly sort the events of a trace, we set the *start timestamp* of these activities equal to the *end timestamp*. Finally, we also added an artificial end and start activity to each trace.

After this filtering process we created four different event logs, one for each process model, in XES format. Table 1 shows, for each prepared event log, the number of activities (*#actitivies*), the number of process instances or tickets (*#traces*), and the total number of events (*#events*) in each event log.

2.2 Conformance analysis

Conformance checking aims to find discrepancies between the modeled behavior, i.e., the process model, and the observed behavior, i.e., the event data. As we have access to the a-priori process models, i.e., the desired behavior, we can conform the recorded behavior of the tickets w.r.t. the a-priori process models and see how many of the handled tickets actually followed the defined steps. The metric used to this end is the *replay fitness*, which measures the extent to which process models can reproduce the traces recorded in the event log. Among the different approaches in the literature related to this particular dimension, we have selected *cost-based fitness metric*, based on *alignments* [15]. An alignment between a trace and a process model is a pairwise comparison between the executed activities and the activities allowed by the model. Such sequences of pairs are called *moves*. Three different moves can be distinguished: *i)* moves only on the event log, i.e., the process model does not allow the execution of a recorded event; *ii)* moves only on the process model, i.e., the process model needs to execute an activity

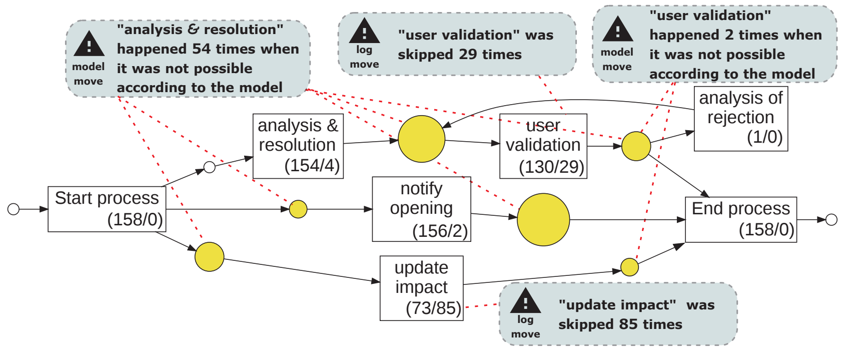
Table 2: Cost-based fitness and number of tickets for the a-priori process models.

	Workflow 1	***Workflow 2***	***Workflow 3***	***Workflow 4***
Cost-based fitness	0.87	0.89	0.96	0.89
#correctly replayed tickets	57 (*36%*)	594 (*51%*)	60 (*71%*)	111 (*18%*)
#incorrectly replayed tickets	101 (*64%*)	557 (*49%*)	24 (*29%*)	500 (*82%*)

not recorded in the event log; and *iii)* moves on both (synchronous moves), i.e., an event in the event log can be correctly replayed through the model.

Table 2 shows the *cost-based fitness* after aligning each event log and process model. For instance, the event log *Workflow 2* has a fitness of 0.89, that is, it can reproduce 89% of the recorded events or, in other words, 11% of the recorded events deviate from the a-priori process model. Additionally, Table 2 also shows the number of tickets that were correctly and incorrectly replayed through their respective process model. In other words, each time a ticket deviates from the a-priori process model, even in a single event, it counts as an *incorrectly replayed ticket*. As can be seen, although the cost-based fitness is relatively high in all cases, there is a significant amount of tickets, in all four event logs, that deviate from the a-priori process model, e.g., 82% of the tickets in *Workflow 4* present a deviation, i.e., an event log/model move.

Through replay fitness techniques we can also obtain a more detailed *local diagnostic*, allowing to detect exactly where the deviations took place. For example, by projecting the alignments between all traces in the event log of *Workflow 1* onto the a-priori process model yields a visualization that shows the location of deviations as shown in the Petri net of Figure 1. In this visualization, each transition shows (in parenthesis) the ratio of synchronous moves (left part) to moves only on the process model (right part). Also, colored places represent *errors* when moves only on the event log occur. Furthermore, the size of the colored places represent the frequency of moves only on the event log. In



model move
"analysis & resolution" happened 54 times when it was not possible according to the model
log move
"user validation" was skipped 29 times
model move
"user validation" happened 2 times when it was not possible according to the model
analysis & resolution (154/4)
user validation (130/29)
analysis of rejection (1/0)
Start process (158/0)
notify opening (156/2)
End process (158/0)
update impact (73/85)
log move
"update impact" was skipped 85 times

Fig. 1: Diagnostic information showing the deviations for *Workflow 1* on the a-priori model.

Table 3: Most skipped and wrongly executed activities for each workflow.

	Workflow 1	***Workflow 2***	***Workflow 3***	***Workflow 4***
Most skipped activity	*update impact* (85)	*warning* (497)	*email to the petitioner* (9)	*priority validation* (421)
Most wrongly executed activity	*analysis & resolution* (54)	*analysis & resolution* (404)	*analyze change order* (13)	*analysis & resolution* (120)

Most skipped activity is related to the model moves, while *Most wrongly executed activity* is related to log moves.

more detail, the deviations in this process model occur in different locations. On the one hand, we detect different model moves: *update impact* was skipped 85 times; *user validation* was skipped 29 times; *analysis & resolution* was skipped 4 times and *notify opening* was skipped 2 times. On the other hand, related to log moves, the activities *analysis & resolution* and *user validation* were executed 54 and 2 times, respectively, when the a-priori process model was not allowing them to happen. Note that, for instance, considering all the possible moves, *user validation* was executed more than 158 times, i.e., the total number of traces. This means that this activity was executed in a loop situation, i.e., it appears more than once within the same trace.

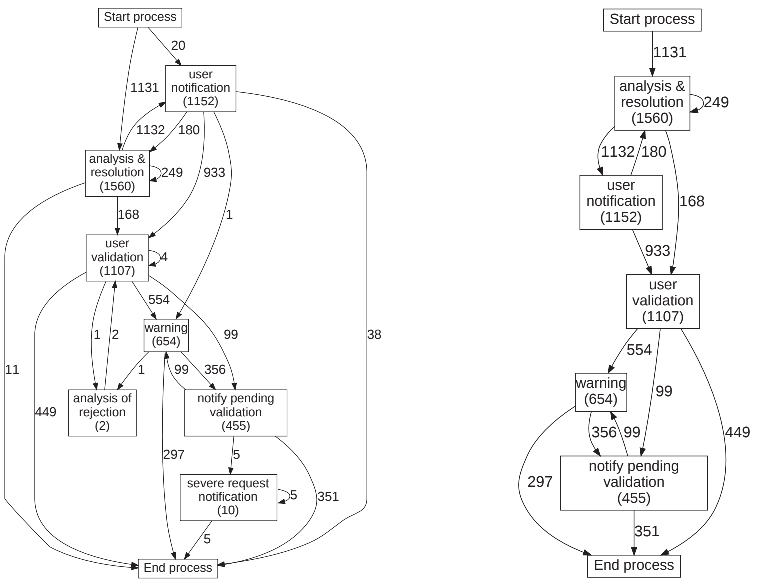
Based on these diagnostics for *Workflow 1*, some of the insights that can be extracted from the previous information are that, for instance, in 53% of the handled tickets, the activity *update impact* was skipped and, in 29%, the activity *user validation* was also not involved, albeit both these activities were designed as required in the a-priori process model. Another deviation is that in 34% of the total tickets, *analysis & resolution* was executed multiple times within the same trace, when this activity was designed to only happen once per process instance. Furthermore, it was also designed to be executed just before *user validation* but, in the previous cases, it was freely executed without any type of restriction, e.g., it was executed just after *user validation.* Table 3 depicts the most skipped (model moves) and wrongly executed (log moves) activities for each process model. Further research will involve finding the reason behind this behavior, following a more *question-driven analysis*, e.g., in which situations is it possible to skip these activities? Is it related to a certain type of tickets?

2.3 Discovery analysis

Following with our exploratory analysis on how this organization is actually handling the different tickets, the next step involves discovering, from the *control-flow* perspective, the real processes based on the recorded events. This analysis usually starts with the visualization of the underlying discovered process model.

Based on the previous conformance checking analysis, all four a-priori models have a fitness over 0.85, i.e., more than 85% of the events happened as planned. Hence, among other indicators such as the number of different process instances, it is quite clear that, from the point of view of process discovery, we are dealing with *Lasanga processes* [1], i.e., the real processes are relatively structured and the cases flowing through such processes are handled in a controlled manner. Hence, it should be possible to discover process models with high(er) values of replay fitness and with a clear structure. Figures 2 and 3 demonstrate that this is, in fact, the described case. More specifically, Figures 2a and 3a show the original discovered process models by ProDiGen for the event logs *Workflow 2* and *Workflow 4*, respectively, in C-net format. A C-net is a graph where nodes represent activities and arcs represent causal dependencies. In this representation there are not places, hence the routing logic is solely represented by the possible input and output bindings. Both these process models reproduce all the behavior of the tickets recorded in the event logs, i.e., they have a perfect replay fitness. Furthermore, in order to focus only on the main behavior of the process models, we also pruned the arcs used less than 5%. Figures 2b and 3b show the resultant pruned models. Additionally, for each process model, we annotated each arc and transition with their frequency of use.

The rather *structured* discovered process models, coupled with their perfect replay fitness, is of special interest to the stakeholders to both detect frequent and infrequent behavior. When these results were presented to the stakeholders, they confirmed that the models representing the most frequent behavior, e.g., the process models in Figures 2b and 3b, were, with slight differences, what they



(a) *Original discovered model.*

(b) *Filtered discovered model.*

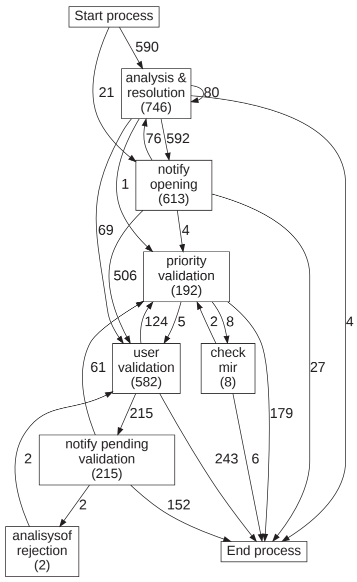
Fig. 2: Annotated c-nets discovered by ProDiGen for the event log *Workflow 2*.

Table 4: Unexpected behavior for each discovered process model.

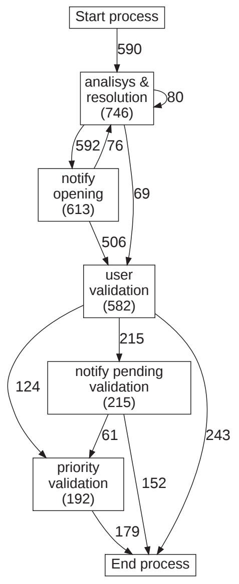
	Workflow 1	***Workflow 2***	***Workflow 3***	***Workflow 4***
Unexpected loops	*analysis & resolution*	*analysis & resolution*	-	*analysis & resolution*
Unexpected skips	*update impact*	*user validation, user notification*	*analyze change order*	*priority validation, user validation*

Unexpected skips and *Unexpected loops* stands for behavior that, in theory, is not allowed, but the discovered process model allows to execute it.

expected. However, they also detected unexpected behavior and deviations in the way of handling certain tickets. Table 4 summarizes the exceptional insights retrieved from this discovery analysis, i.e., loops that in theory are not allowed (*unexpected loops*), or activities that are mandatory but in reality can be skipped (*unexpected skips*). Among the different deviations detected by the stakeholders, it is noteworthy, in both processes (Figures 2b and 3b), the high frequency of the self-loop in the activity *analysis & resolution*. Additionally, also in both these processes, the activity *user validation*, in theory mandatory, can be skipped in the discovered model. In other words, this means that there were tickets that completed without any kind of *validation*.



(a) *Original discovered model.*

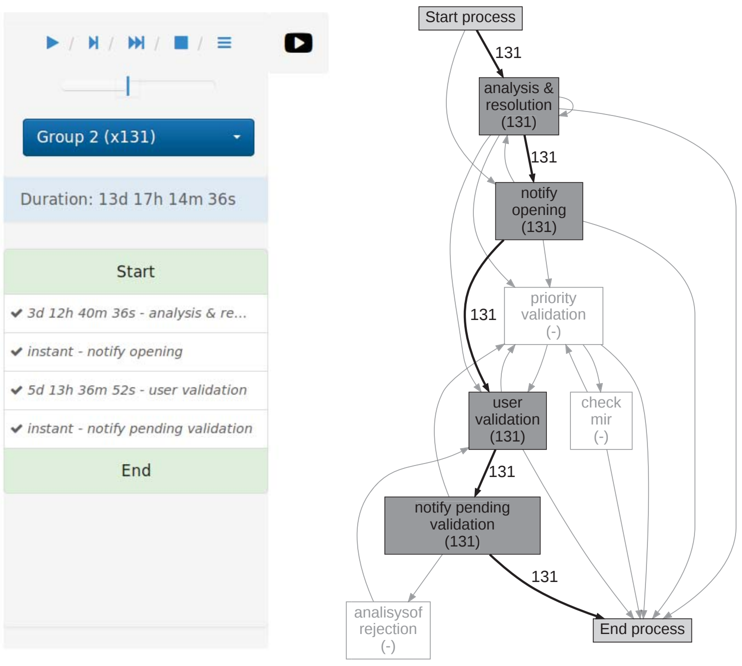


(b) *Filtered discovered model.*

Fig. 3: Annotated c-nets discovered by ProDiGen for the event log *Workflow 4*.

ProDiGen platform provides a visual interface where stakeholders can reproduce, through the discovered process models, the actual path of each trace. This was crucial to retrieve valuable information about the behavior of the whole process. Figure 4 shows a snapshot of this *process player*, on the event log *Workflow 4*. This *process player* represents a set of controls to reproduce the event log over the process model. In this example, we have grouped the traces that followed the same path in the process model. Hence, in this visualization we show the path —the dark gray activities and arcs— followed by the traces grouped in *Group 2*, i.e., 131 different traces that share the same sequence of activities. Furthermore, this player also shows different performance statistics (this kind of analysis is extended in more detail in Section 2.4), in the left part of Figure 4, such as the average completion time of this group of traces, i.e., 13 days, or the average completion time of each activity considering only this group of traces, e.g, *user validation* took 5 days on average within these 131 traces. Additionally, this player also enables to reproduce specific traces, allowing to gain more fine grained insights, in a visual way, into how, who and when an specific trace was executed through the discovered process model.

Furthermore, through this discovery analysis we obtained valuable feedback from the people involved in the process on how to improve the mining of these kind of processes in further analysis. Specifically, the timestamp is of particular



Duration: 13d 17h 14m 36s

Start

✓ 3d 12h 40m 36s - analysis & re...

✓ instant - notify opening

✓ 5d 13h 36m 52s - user validation

✓ instant - notify pending validation

End

Fig. 4: Snapshot of the *process player*, in ProDiGen platform, on the process model discovered for the event log *Workflow 4*.

value when mining these kind of event logs. For instance, when the same activity is executed multiple times in a short period of time, it should be considered as the same activity. Additionally, when two different activities are executed sequentially in a short period of time, it should be considered that they are executed in parallel regardless if they always appear as a sequence.

2.4 Performance analysis

The last part of this case study relies on the performance analysis. Process mining provides a wide range of performance techniques [1]. Among them, the dotted chart is one of the most powerful tools to view a process from different angles. Concerning this paper, we use the dotted chart to gain an overall view of the performance of the event log *Workflow 2*. Figure 5 shows this dotted chart. In this chart, time is measured along the horizontal axis, and each trace represented along the vertical axis where each dot is an event. The color of each dot represents the activity of the process, e.g., the red dots represent the activity *user validation*. Note that, in this visualization, we omitted the artificial start and end activities.

Based on this dotted chart of the event log, different observations can be made. At first glance, we can see that the process does not follow a constant arrival of tickets, i.e., the initial events of all traces do not form a straight line: there is a clear difference between the influx of tickets before and after 2014. More specifically, we can see that there is a significant increase in the arrival rate of tickets after the last months of 2014. Moreover, this arrival is quite steady, i.e.,

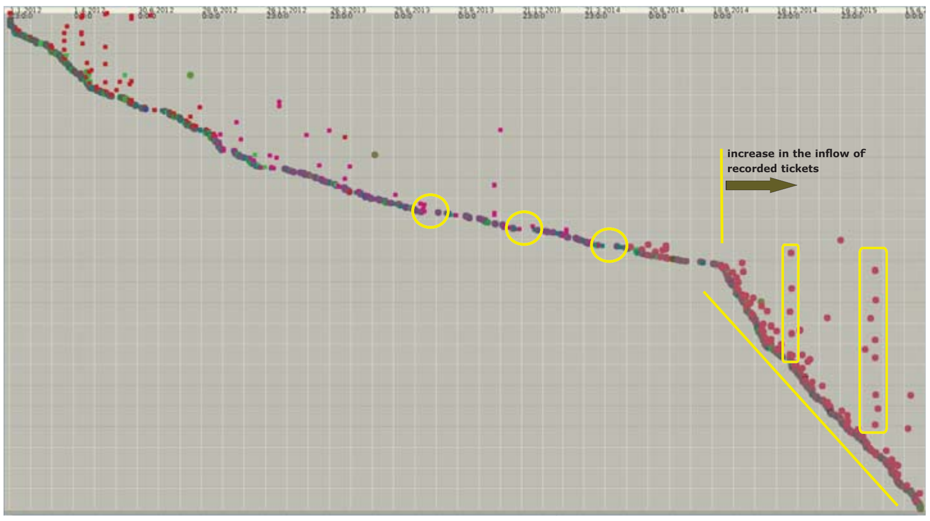


Fig. 5: Dotted chart, retrieved with ProM, for the event log *Workflow 2* using absolute (real) times for the horizontal axis. Each horizontal division represents a calendar month. The vertical axis represents the traces.

we can draw a rather straight line alongside the initial events during this period of time (bottom right of Figure 5), i.e., from 2014 onwards. Additionally, it seems that, in some situations, certain sets of activities are executed in batches, i.e., the same activities are executed for different cases in the same interval of time. For instance, the activities inside the colored boxes (bottom right of Figure 5) seem to show this particular behavior. We can also notice that there are periods of inactivity (some of them are marked with the colored circles in the middle of Figure 5), where no events were recorded.

On the other hand, for some tickets, events are recorded a long time after their arrival, whereas for the majority of the tickets most events are observed in the first couple of days. Figure 6 shows a better view of this behavior. In this figure, we use the relative times, i.e., all the tickets start at time zero, and they are sorted in descending order by their real duration. As can be seen, most of the cases ended within the first days, or even in the same day, of being registered in the organization. However, we can find different tickets that took much longer than expected, e.g., more than 10 days. Furthermore, we can see exceptional cases that took even more than a year (the bottom of Figure 6). In general, it seems that when the ticket goes through the activities *user validation* and *warning*, i.e., the events represented by a red and pink dot, respectively, the time to handle a ticket increases.

As shown, it is possible to divide the real process of *Workflow 2* in two time intervals, based on the inflow of tickets: before and after the increase in the arrival rate of tickets in 2014. Furthermore, we detected the same pattern in the other three event logs within the organization. With this in mind, an interesting

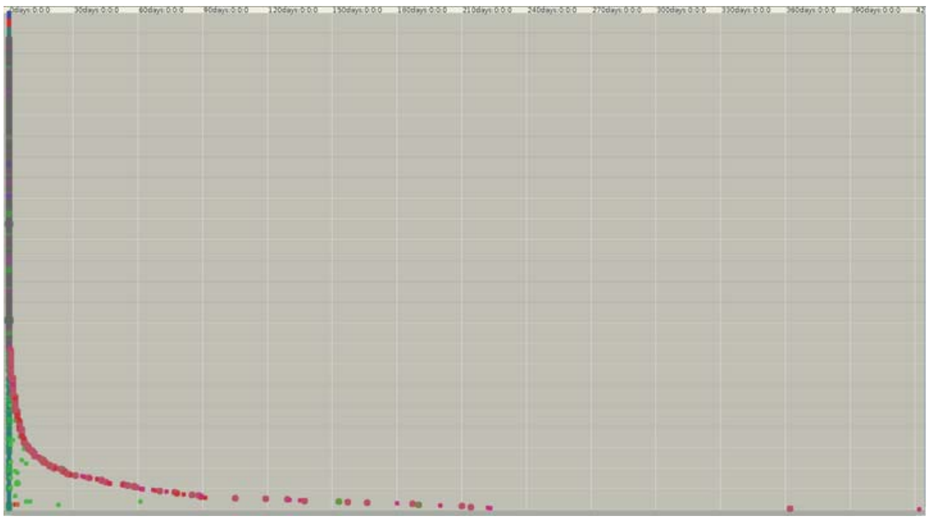
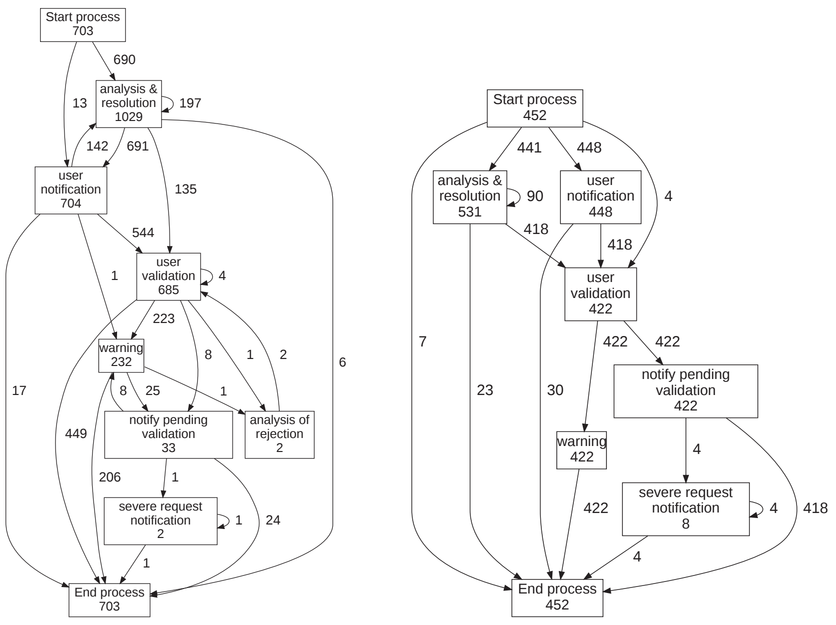


Fig. 6: Dotted chart, retrieved with ProM, for the event log *Workflow 2* using relative times for the horizontal axis, i.e., all traces start at time 0. Each horizontal division represents 30 days. The vertical axis represents the traces.

analysis would involve discovering the process models in these two time intervals to scrutinize how the process behave under such conditions. Hence, we created two different logs with the tickets before and after 2014 for *Workflow 2*, more specifically, before and after September 2014. Figure 7 shows the discovered process models for these two time intervals. Again, both solutions have a perfect replay fitness, i.e., they reproduce all the recorded behavior in both event logs. As can be seen, the process model describing the behavior of the tickets recorded after 2014 (Figure 7b) is more well-structured than the process model discovered for the tickets recorded before 2014 (Figure 7a). Furthermore, the process model of Figure 7b, i.e., the modeled behavior after 2014, describes, with slight differences, the expected behavior designed by the stakeholders. However, the discovered model of Figure 7a, i.e., the modeled behavior before 2015, depicts more deviations and unexpected behavior. In other words, after September 2014, there was an significant improvement on the way of handling the different tickets through this particular *workflow*, with less deviations, i.e., tickets that followed not defined rules from the desired process model. Remark that, after splitting the behavior in all the remaining event logs, we found the same behavior in the discovered process models, i.e., after September 2014, the way of handling



(a) *Discovered process model for the tickets recorded before September 2014.*

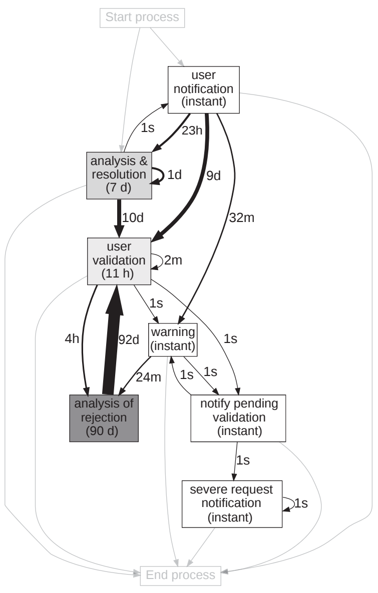
(b) *Discovered process model for the tickets recorded after September 2014.*

Fig. 7: Annotated c-nets discovered by ProDiGen for the event log *Workflow 2* before and after September 2014.

the different tickets was followed in a more strict way, as stated in the different designed process models.

Performance analysis can also be achieved by enhancing process models with, for instance, the time attributes of the events recorded in the event log. In other words, using the previously discovered process models in Section 2.3, and the timestamp of the events, it is possible to detect bottlenecks and other types of behavior that could negatively affect the whole performance of the process. Figure 8 shows the throughput of the process model discovered in Section 2.3 for the event log *Workflow 2*. This model was extended with the timestamp for both the activities, and the *layover* between them, i.e., the time between the *completion* of the preceding activity and the *start* of the next activity. In this process model, each arc is annotated with the *average* time between activities. Moreover, the darker it is in comparison with the rest of the arcs, the longest is the time between two activities. The same annotation is used regarding the time of the activities.

As can be seen, most of the *layovers* between activities are almost automatic, taking a second or less. However, there is one *layover* that stands out from the rest: the arc from *analysis of rejection* to *user validation*, which takes 92 days. Concerning the activities, it is possible to identify several of them that are automatic, i.e., they are *instant*. However, we also find an activity that took, on average, 90 days: *analysis of rejection*. From a global perspective, if we analyse



Start process
user notification (instant)
1s
23h
analysis & resolution (7 d)
1d
9d
32m
10d
user validation (11 h)
2m
1s
4h
92d
warning (instant)
1s
24m
1s
1s
analysis of rejection (90 d)
notify pending validation (instant)
1s
severe request notification (instant)
1s
End process

Fig. 8: C-net discovered for the event log *Workflow 2*, extended with the time perspective.

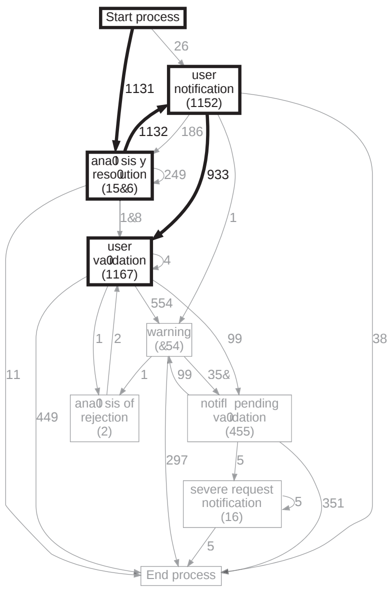


Fig. 9: Most frequent pattern within the process model discovered for the event log *Workflow 2*.

the frequency of use of this part of the model (Figure 2a), *analysis of rejection* was only executed two times in the whole process, but being very time consuming in the whole process performance.

Within the visual interface provided by ProDiGen, it is also possible to detect the most frequent patterns in a process model, allowing to visualize the critical parts. These patterns are extracted using both the information of the the event log and the discovered process model to extract the frequent patterns. Hence, using this algorithm it is possible to detect subprocesses (both activities and control structures) that can be replaced with high level activities to, for instance, reduce the complexity of a process model. For instance, Figure 9 highlights the frequent pattern within the process model discovered for *Workflow 2*, with a threshold above 70%, i.e., the pattern must be fulfilled more than 70% of the times the process was executed. In this particular case, the most frequent pattern in the whole model is a sequence of $\langle$*analysis & resolution* $\rightarrow$ *user notification* $\rightarrow$ *user validation*$\rangle$, with a frequency of 81%, i.e., 81% of the behavior recorded in the event log went through this pattern. This visualization allows us to easily check which part of the model is the most congested.

3 Conclusions

We have presented a case study of applying process mining on a real IT service management scenario. The real data set has 2,004 requests and incidents recorded between 2012 and 2015 within an organization. First, we extracted and prepared the data. Then, based on the a-priori models, we performed a conformance checking analysis, followed by the discovery of the real process models. Finally, we measured the actual performance of the processes. In this paper, we focused on a *data-driven* project, providing valuable insights about the way of handling the different tickets. For instance, the change of behavior in the global processes before and after 2014. Other insights are related to how, for some tickets, it was necessary to *redo* certain activities, related to wrongly assigned tickets, or, for other tickets, that there was no validation and/or notification to the user and/or staff. Regarding the latter, this behavior was due to anticipated cancellations that were not properly recorded in the system, leading to open tickets, or unexpected endings. Further analysis will involve giving answer to the untapped questions, such as: the analysis of the tickets that were involved in more than one process model, the *handover* of work, etc.

Acknowledgments. This research was supported by the Spanish Ministry of Economy and Competitiveness (grant TIN2014-56633-C3-1-R, co-funded by the European Regional Development Fund - FEDER program), the Galician Ministry of Education under the projects EM2014/012, CN2012/151, GRC2014/030.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. 1st edn. Springer (2011)

2. Mans, R.S., Schonenberg, M.H., Song, M., van der Aalst, W.M.P., Bakker, P.J.M.: Process mining in healthcare - A case study. In: Proceedings of the First International Conference on Health Informatics, HEALTHINF. (2008) 118–125
3. Partington, A., Wynn, M., Suriadi, S., Ouyang, C., Karnon, J.: Process mining for clinical processes: Acomparative analysis of four Australian hospitals. ACM Transactions on Management Information Systems **5**(4) (2015) 1–19
4. Forsberg, D., Rosipko, B., Sunshine, J.L.: Analyzing PACS usage patterns by means of process mining: Steps toward a more detailed workflow analysis in radiology. Journal of Digital Imaging **29**(1) (2015) 47–58
5. Sztyler, T., Völker, J., Carmona, J., Meier, O., Stuckenschmidt, H.: Discovery of personal processes from labeled sensor data – An application of process mining to personalized health care. In: Proceedings of the 2015 International Workshop on Algorithms & Theories for the Analysis of Event Data, ATAED. Volume 1371 of CEUR. (2015) 22–23
6. Sedrakyan, G., Weerdt, J.D., Snoeck, M.: Process-mining enabled feedback: “tell me what I did wrong” vs. “tell me how to do it right”. Computers in Human Behavior **57** (2016) 352–376
7. Vazquez Barreiros, B., Lama, M., Mucientes, M., Vidal, J.: Softlearn: A process mining platform for the discovery of learning paths. In: Proceedings of the 14th International Conference on Learning Technologies ICALT. (2014) 373–375
8. Park, M., Song, M., Baek, T.H., Son, S., Ha, S.J., Cho, S.: Workload and delay analysis in manufacturing process using process mining. In: Proceedings of the 3rd Asia Pacific Conference on Asia Pacific Business Process Management, AP-BPM. Volume 219. (2015) 138–151
9. Stolfa, J., Kopka, M., Stolfa, S., Kobersky, O., Snásel, V.: An application of process mining to invoice verification process in SAP. In: Proceedings of the 4th International Conference on Innovations in Bio-Inspired Computing and Applications, IBICA. Volume 237 of Adv. Intell. Syst. Comput. (2014) 61–74
10. Spagnolo, G.O., Marchetti, E., Coco, A., Scarpellini, P., Querci, A., Fabbrini, F., Gnesi, S.: An experience on applying process mining techniques to the tuscan port community system. In: Proceedings of the 8th International Conference on Software Quality. The Future of Systems-and Software Development, SWQD. Volume 238 of LNBIP. (2016) 49–60
11. Weerdt, J.D., Schupp, A., Vanderloock, A., Baesens, B.: Process mining for the multi-faceted analysis of business processes – A case study in a financial services organization. Computers in Industry **64**(1) (2013) 57–67
12. van Eck, M.L., Lu, X., Leemans, S.J.J., van der Aalst, W.M.P.: PM^2: A process mining project methodology. In: Proceedings of the 27th International Conference on Advanced Information Systems Engineering, CAiSE. Volume 9097 of LNCS. (2015) 297–313
13. Vázquez-Barreiros, B., Mucientes, M., Lama, M.: ProDiGen: Mining complete, precise and minimal structure process models with a genetic algorithm. Information Sciences **294** (2015) 315–333
14. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: XES, XESame, and ProM 6. In: Information Systems Evolution. Volume 72 of LNBIP. (2011) 60–75
15. Adriansyah, A.A.: Aligning observed and modeled behavior. PhD thesis, Technische Universiteit Eindhoven (2014)