Investigations of Process Mining Methods to discover Process Models on a Large Public Administration Software

Florent Mouysset, Célia Picard, Christophe Bortolaso, Frédéric Migeon, Marie-Pierre Gleizes, Christine Maurel, Mustapha Derras

Investigations of Process Mining Methods to discover Process Models on a Large Public Administration Software

Florent Mouysset[1,3], Célia Picard[2*], Christophe Bortolaso[1], Frederic Migeon[3], Marie-Pierre Gleizes[3], Christine Maurel[3] and Mustapha Derras[1]

1. Berger-Levrault, 64 Rue Jean Rostand, 31670 Labège, France
{first name}.{last name}@berger-levrault.com

2. ENAC, 7 Avenue Edouard Belin, 31400 Toulouse, France
{first name}.{last name}@enac.fr

3. IRIT, Université Toulouse 3 (Paul Sabatier),31400 Toulouse, France
{first name}.{last name}@irit.fr

ABSTRACT. *Maintaining large and aging applications in a software house, with heterogeneous technologies, is very challenging. Whereas it is mandatory to continuously enhance user experience and maintain a good quality of service, the real business usage can be difficult to know precisely. To reach this goal, our project is to discover business models from the analysis of "logs". In this paper, we report on existing studies about applying process mining techniques and on our own experience with large datasets generated from daily end-user activities within an existing public services software. Our experiments led us to identify an interesting combination of features making our data hard to process with existing techniques. We conclude by providing perspectives to enable process discovery with such specific data.*

KEYWORDS: *Software logs analysis, Interweaving, Process Mining*

1. Introduction

The complexity of software for public services increases with the constant evolution of regulations and with users' requirements. This has major impacts on software quality and maintenance processes. First, on the software editor side, this usually leads to an increasing number of bugs (Subramanyam & Krishnan, 2003). The scarceness of users' feedbacks collected by support teams, the heterogeneity of execution contexts and data make the bugs very hard to reproduce and to fix. As the project team designs the tests a priori, they may not reflect the users' practices. Thus, a gap appears between the theory and the reality (second Lehman's law (Lehman, 1980)), leading to failures on untested and unexpected cases. Then, from the user's point of view, the software becomes more difficult to use, interact with

* The work described in this paper was realized during Célia Picard's time at Berger-Levrault's.

and understand (Thompson *et al*., 2005). A new paradigm emerges and tries to design intelligent software, able of adapting themselves to specific users (Salinesi, 2017). This requires observing, collecting and understanding the users' activities. Logger systems constitute good solutions to collect users' activities. We made the assumption that logger systems combined with process mining techniques (Ailenei *et al.*, 2011) would enable the discovery of the observed system model. The discovered business process model would allow us, in a first step, to elicit use cases like user goals or business scenarios. In a second step, it would lead to prioritization of the maintenance, more realistic tests, and business intelligence.

In this paper, we report on experiments and lessons learned from applying process mining techniques on logs extracted from an existing on-use large public administration software. Our work can be compared to Astromskis's (Astromskis *et al*, 2015) who managed successfully to use process mining to understand user interaction on software. It led us to believe that their approach would be easily replicable on our dataset.

In the following pages, we first present our case study, the logged data and the associated software. Then, after presenting a general review of existing approaches, the best technique is selected to conduct experiments. Section 4 details these experiments. To conclude the paper, we stress that the presence of certain characteristics in our logs makes the process mining techniques ineffective. The problem is conceptual and not related to the logging system.

2. Case Study Description and Associated Data

We collected logs[1] generated by a rich client application used by town hall agents that we will call here "Software for public services" (SFPS). This software is composed of four modules. In this study, we focus on the analysis of logs coming from the Civil Status Management Module (hereafter called CSMM). This module contains numerous features related to civil registries, such as elections, births, deaths or court jurors. The software interface is composed of multiple tabs, each of them containing multiple forms. The navigation inside the CSMM is structured as follows: it starts with a home page containing a list of buttons. When clicking a button in this menu, a new tab opens and gets the focus. This new tab provides access to a second level menu containing itself user interface components to access forms and functional features.

The logging system is implemented in a software layer shared by all the different modules. This enabled developers to rely on a generic tracing system and to focus only on the development of functional features.

Despite the simplicity of the main navigation, the CSMM is extremely large and dense. Indeed, it is composed of about 600 different forms representing about

[1] Data are on open access on https://github.com/FM-BL/PublicCSMM_Logs

200,000 lines of code. We collected logs from 104 users over one year. The data contains 227,782 events for 60 mega-bytes in an XML structure.

```
<ExportList>
    <FormName>A</FormName>
    <FormDescription>A long description</FormDescription>
    <UserID>USER1</UserID>
    <OpeningTimestamp>2016-07-07T09:00:00</OpeningTimestamp>     A1
    <ClosingTimestamp>2016-07-07T10:00:00</ClosingTimestamp>     A2
</ExportList>
<ExportList>
    <FormName>B</FormName>
    <FormDescription>B long description</FormDescription>
    <UserID>USER1</UserID>
    <OpeningTimestamp>2016-07-07T09:15:00</OpeningTimestamp>     B1
    <ClosingTimestamp>2016-07-07T09:30:00</ClosingTimestamp>     B2
</ExportList>
<ExportList>
    <FormName>C</FormName>
    <FormDescription>C long description</FormDescription>
    <UserID>USER1</UserID>
    <OpeningTimestamp>2016-07-07T09:30:01</OpeningTimestamp>     C1
    <ClosingTimestamp>2016-07-07T09:59:59</ClosingTimestamp>     C2
</ExportList>
<ExportList>
    <FormName>D</FormName>
    <FormDescription>D long description</FormDescription>
    <UserID>USER1</UserID>
    <OpeningTimestamp>2016-07-07T09:35:00</OpeningTimestamp>     D1
    <ClosingTimestamp>2016-07-07T09:55:00</ClosingTimestamp>     D2
</ExportList>
<ExportList>
    <FormName>E</FormName>
    <FormDescription>E long description</FormDescription>
    <UserID>USER1</UserID>
    <OpeningTimestamp>2016-07-07T10:15:00</OpeningTimestamp>     E2
</ExportList>
```

Figure 1. Example of CSMM log

Each event represents the opening and the closing of a form. As depicted on Figure 1, the log is composed of a sequence of events each called "*ExportList*". Each event has five fields: "*FormName*" represents a unique form code and "*FormDescription*" contains a longer and more readable description; "*UserID*" indicates the user id performing the action; "*OpeningTimestamp*" and "*ClosingTimestamp*" are the form opening and closing dates.

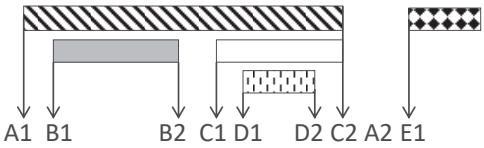

Figure 2. Temporal representation of the Figure 1 log

Figure 2 illustrates on a timeline the sequence of opening and closing events of the five forms present in Figure 1. Here, the form A enables to reach form D through form C. The closure of forms works with a FILO (i.e. First In Last Out) behavior.

However, the software interface is designed to allow other additional navigation instances. From a form D another form C can be opened in a new tab, both being

independent. Thus, it is possible to close the form D first, and after form C. Closing forms can also work with a FIFO (i.e. First In First Out) behavior. This indicates that a form closure does not necessarily imply a parent relationship. Furthermore, in some cases, the closing field ("*ClosingTimestamp*") might be missing (form E does not have a closing tag). It means either that the form encountered an error, making it impossible to write the closing date or that the form was not closed before leaving the application. Finally, we observed that sometimes, even if two forms are dependent, the closing date might be incoherent. For instance, the form C is child of form A (and are dependent, no new navigation instance) but the closing of form C occurs after the closing of its parent (A2). These three scenarios illustrate the wide range of behaviors that can be found in the processed data in our case study. The next section describes the different existing approaches that use input logs to discover process models.

3. Discovery Process: Existing Approaches

Multiple scientific domains focus on the analysis of human activity logs. In fact, several methods and algorithms designed to discover usage patterns and process models on business activity logs exist. We identified two types of relevant paradigms: 1) web mining and 2) process mining.

3.1. Web Mining

The web mining or web usage mining offers several approaches to analyze a web application and its usages (Srivastava *et al.*, 2000). Usually, the techniques are user-centric: all the events are related to a user. Existing techniques are most of the time based on sequence mining and pattern discovery methods. One of the typical purposes of web mining is to find the main way to accomplish a predefined goal. For instance, this purpose could be "What is the most frequent path to buy a specific product in an online shop?" With this kind of information, it becomes possible to optimize navigation paths and hence, increase sales (Spiliopoulou *et al.*, 2000). The goal of web mining differs from ours even though the data are similar. In web mining, the user is linked to a web session and each web session corresponds to a single use case (Srivastava *et al.*, 2000). Moreover, in the analysis the user goal is usually known beforehand: a purchase, a web page access or more generally the achievement of a well-known special action.

In our case, we do not have any information about users' goals in the current navigation path. Worse, the user can perform several use cases (i.e. pursue several goals) concurrently and we do not know how to relate the events to use cases. Thus, these techniques appear irrelevant for our problem.

3.2. Process Mining

Compared to the specificities of web analytics and mining, process mining appears to be a more suitable approach to our problem. The process mining is a recent scientific domain defined as a mix between data mining and workflow

analysis (van der Aalst, 2016). The process mining grasps three main aspects: 1) process discovery, 2) conformance model and 3) process enhancement. Since we seek to produce a process model from logs, we focused on the first aspect of the process mining: the process discovery. The process discovery is a collection of techniques that take logs as inputs and give process models as outputs. The input logs contain traces. Each trace is a sequence of events referring to the same case. Events are footprints left by the users when performing a business activity. Each event contains, at least, the name of the process activity, a timestamp and a specific process instance identifier[2]. Generally, the inputs are process-oriented (Pourmasoumi & Bagheri, 2017). This means that each trace corresponds to a process execution. Each process execution is identified by a unique id called the process instance id (or case id). The events belonging to the same process execution share the same process instance id: they are labelled. Logs containing labelled events are called labelled logs. The logs can contain various sorts of noises ((van der Aalst, 2016) p.148-151) and some data can be missing, incorrect or imprecise. This problem can occur in a continuous, intermittent or unpredictable manner. Business process models are generated as output. Various formalisms exist such as Petri Nets and Business Process Models and Notations (BPMN). Quality metrics can also be computed on these models. The most frequent metrics are the fitness (i.e. the ability to replay the behaviors seen in the logs) and the precision (i.e. the ability to forbid the behaviors unseen in the logs).

3.2.1. Approaches dealing with labelled logs

To enable process discovery, Cook and Wolf early proposed solutions based on neural-networks or Markov models, each solving a limited part of the problem. For example, the KTail approach finds a correct model but is noise-sensitive. On the contrary, the RNet methods find a less accurate model but are robust to noise (Cook & Wolf, 1998). However, none of those methods can manage concurrency aspects: in a business process, some activities can be performed simultaneously, and it is important to detect and represent these concurrent activities (Buijs *et al.*, 2012). The α-algorithm (van der Aalst *et al.*, 2004) brings a formalism to discover WorkFlow nets (WF-nets) (van der Aalst, 1998). WF-nets are a subclass of Petri Nets that can offer some interesting properties like soundness and safeness. They provide very good results, but these methods remain extremely noise-sensitive (van der Aalst *et al.*, 2004).

Probabilistic and statistical approaches are more robust and reliable than the α-algorithm (van der Aalst, 2016). The Heuristics Miner Algorithm (Weijters *et al.*, 2006), brings a significant improvement by solving problems such as the short loop discovery or the mining of long-distance dependencies (Wen *et al.*, 2006). Günther developed "The Fuzzy Miner" and provided techniques to discover processes from noisy data by using a graph model formalism (Günther & van der Aalst, 2007). With two new defined metrics (the significance and correlation) the method captures the

[2] In any case, an id is a unique identifier.

main patterns. But human intervention is required to tune parameter settings and identify the best model. Overall, probabilistic and statistical approaches are adapted to simple events but are limited when data is rich and complex.

The last family of approaches studied is the genetic algorithms. For instance, Alves de Medeiros's work (Alves de Medeiros, 2006) proposes to reuse the main concept of the genetic algorithms in process mining. With her method, an individual is considered as a process model on which selection and reproduction steps are applied. The main drawbacks of genetic algorithms are the long execution time and the local minimum risk. Moreover, genetic algorithms generate very complex models when using real-life logs (De Weerdt *et al.*, 2012).

3.2.2. Approaches dealing with unlabeled logs

Because obtaining labelled logs can be very challenging (Pérez-Castillo *et al.*, 2013), various methods try to deal with unlabeled data. The solution relies on labelling events artificially. We have identified two approaches: 1) making assumptions about structural, behavioral and temporal aspects of logged activities (Pourmirza *et al.*, 2015; Walicki & Ferreira, 2011), and 2) techniques based on experts' knowledge used to discover correlation rules and group events into traces (Pérez-Castillo *et al.*, 2013).

Pourmirza (Pourmirza *et al.*, 2015) proposes an algorithm to label logs coming from orchestration services. Three conditions must be respected: 1) it does not work with data coming from other services than acyclic orchestration services, 2) metrics on duration per activity must be given to the algorithm, and 3) the idle time between two requests should not be equal to the activity duration. Similarly, Walicki and Ferreira propose a probabilistic approach (Walicki & Ferreira, 2011) to process discovery with unlabeled logs. The approach has a high level of abstraction because the problem can be reduced to sequence mining challenges. Their technique tries to cover the sequence with a minimal set of patterns. However, some conducted experiments show that this approach is not suitable to process large logs (Walicki & Ferreira, 2011).

When the assumptions about data and processes are too variable, experts can explicitly write rules to inform the tagging algorithms. Pérez-Castillo (Pérez-Castillo *et al.*, 2013) propose a semi-automatized process able to build correlation rules. An event must contain various attributes and the logging system must be designed to put the candidate correlation attributes into events. An expert can choose the attributes in the log and determine the correlation rules.

3.3. Synthesis on process discovery

Broadly speaking, the literature shows that a large variety of techniques are available. Web mining is hard to apply because our users' goals are unknown. It is unclear in our software when and where use cases start and end. We chose to skip the early approaches of process mining for the more efficient recent ones. Process mining approaches dealing with unlabeled logs are not suitable as they require realistic temporal and structural assumptions. As described before, due to the

activity interweaving and interruptions, it is impossible to make any realistic statement about the users' activities in our logs. Our application is very large and has many legacy features; it is impossible to find experts with enough business knowledge and time to choose the attributes for each code injection. The difficulties of improving logging system are detailed in section 6.3. The logs only represent the opening and closure of forms, tabs and windows. The missing of the case id is the only limitation that forbids us to apply approaches dealing with labelled data. Thus, we artificially labelled our events to execute existing process mining techniques on them, as described on Astromskis's study (Astromskis *et al.*, 2015).

4. Experiments with Process Mining

Based on our data and literature review, process mining methods appeared to be promising approaches to process our logs and generate the software business process model. To generate the business process model of CSMM, we have conducted experiments with several process mining techniques implemented in ProM. ProM is an academic process mining software that offers several process mining techniques through plugins.

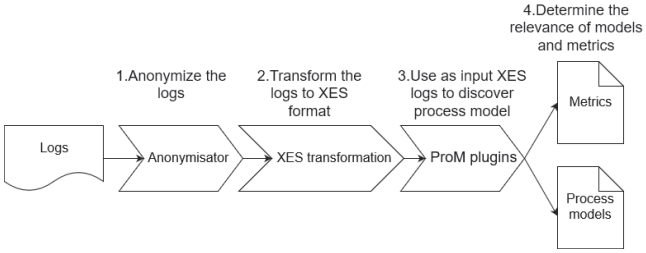

Figure 3. The main steps of our case study experimentation

We tested four different methods aiming to discover valid process models and compared the results. Hereafter, we describe our protocol and data transformation process, the tools, and finally our results.

4.1. Protocol

As described in Figure 3, our protocol is divided into three main steps: 1) anonymization, 2) XES transformation and 3) execution of ProM.

1. *Anonymization.* Due to privacy concerns, the first step consists in removing data that can be related to individuals such as: emails, addresses, names, surnames, etc. This information is replaced by IDs to maintain coherence and integrity in the logs. This process is reproducible and non-reversible.

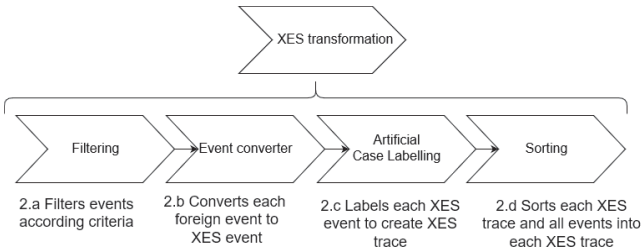

Figure 4. Sub-steps of the XES transformation

2. XES transformation. ProM accepts XES files as inputs. Therefore, a transformation from "CSMM" logs format to XES format is required. Labelling the events can be very difficult, thus we developed several strategies based on a successful study (Astromskis *et al.*, 2015). Each of them provides a smaller granularity supposed to give a better trace. We tested different transformations methods in the next sub-sections. Each concern is addressed as a sub-step as depicted in Figure 4.

2.a Filtering: we removed some noise and execution errors from the raw logs. More precisely, we identified and removed the events where the closing date is missing as they indicate a software failure, according to the development team. Overall, 7939 events were removed, corresponding to 3.3% of the total dataset.

2.b Event converter: various alternatives are available to translate CSMM events into XES events, each providing different advantages and semantics. We identified two strategies:

1) "1EGRC1XES" strategy consists in mapping each field of the CSMM event to the corresponding field of the XES event. This is the straightforward approach.

2) "1EGRC2XES" strategy considers that an XES event indicates either a form opening or a form closing. Thus, the conversion may produce two XES events for each event, one as a "start life-cycle transaction", the second as a "complete life-cycle transaction".

2.c Artificial Case Labelling: XES logs are structured into traces, which is a missing concept in our logs. In the XES format, a trace reflects a case which is not delimited in our data. Thus, we tried five different strategies to provide each event with a case id and artificially recreate traces from our logs:

1. The "Naive" strategy considers that all events have the same case id. Hence, only one trace is created containing all the events.
2. The "User Trace" (UT) strategy builds one trace per user. All events performed by the same user have the same case id.
3. The "User Trace by Day" (UTD) strategy considers the day scale and the user labelling; a case represents a user's daily activities. Hence, all the events performed by the same user in the same day have the same case id. Note that a case

cannot be performed over several days because town hall agents must respect working hours. Moreover, none of the use cases require several days to be accomplished.

4. The "User Home Trace by Day" (UHTD) considers the home page visit as the end of the previous business scenario and the beginning of a new one. All the events performed by the same user in the same day between two home page accesses have the same case id.

5. The "User Sub-Home Trace by Day" (USHTD) is based on the same principle as UHTD but considers the direct children forms of the home page. CSMM counts 11 sub-forms at a 2nd level of navigation. When an opening event of a sub-home form occurs, all the encountered events form a trace. All the events performed by the same user in the same day between sub-home forms have the same case id. We chose to stop at the second level because it still represents a manageable size of sub-nodes. The number of forms at the 3rd level increases exponentially.

We crossed all the possibilities between our two event conversion strategies and our five artificial use case labelling strategies. We decided not to consider the combination of USHTD and 1EGRC2XES strategies. In fact, we observed that by applying this strategy 50% of the events were located between accesses of forms. This led us to believe that this strategy would not provide any significant results. The artificial use case labelling strategies can be considered as cutting functions. In fact, all the opening forms of a considering level are cut points. The forms path before the cut belongs to a use case; the forms path after the cut belongs to another use case. We restricted our study to these five strategies because more refined approaches would over-cut the logs and would have created too many traces.

2.d Sorting: Finally, we sort the events by date in the resulting XES files. In the 1EGRC1XES condition, we sort the events by start date (i.e. opening of the form). In the 1EGRC2XES condition, we sort the events by the date kept in the event, thus opening date of the form for some events and closing date of the form for the others.

Execution. The last step consists in executing the four plugins, described in the apparatus below, on various data sets. Exactly, each one of the four listed plugins are tested with each generated XES file using the two different converters (2b) and the 5 different labelling strategies (2c), hence, a total of 4*2*5=40 experimentations.

4.2. Apparatus

To perform process mining, we rely on ProM v.6.7. ProM is an academic process mining software[3]. Various plugins exist, especially for process model discovery. As explained above we aim at testing and comparing four different methods on our data (see Table 1). The experiment has been carried on a standard laptop machine including 16Gb with 6Gb RAM dedicated for JVM, Intel® Core™ i5-4210M CPU@ 2.60GHz. Our evaluation is based on the replay fitness metric (Buijs *et al.*,

[3] http://www.promtools.org/doku.php

2012). The replay fitness indicates how much the discovered model can reproduce the behavior capture in logs. The replay fitness is ranged between 0 (the discovered model cannot replay any part of the logs) and 1 (a perfect replay).

Table 1. ProM Plugins Used and Tested

Name	Version	Output
HeuristicsMiner	6.7.70	Fitness metrics
Fuzzy	6.7.53	Replay percentage
Alpha Robust Miner	6.7.70	Petri Net. PN Conformance Analysis plugin to compute conformance
Evolutionary Tree Miner (ETM)	6.7.168	Process Tree / Display fitness per generation

4.3. Results

The results of the experiments are listed on Table 2. After the XES transformation step, we obtain traces corresponding to process runs in process mining. The trace number increased as the trace granularity became smaller. The event conversion produces a total of 227,782 events, considering only the opening of the forms (1EGRC1XES), and logically twice as much (455,564) when also considering closing events (1EGRC2XES). Overall, the results are disappointing, and no method allow us to discover a reliable process.

First, surprisingly, with the Heuristic Miner, the more we try to label use cases precisely, the lower the fitness is. We also observe that in over one-third of our tests, the Heuristic Miner could not provide positive fitness. Negative fitness indicates a very low success due to many remaining tokens in the associated Petri Net. The "1EGRC1XES" / "User Trace" condition provided the best result with a fitness of 0.55. For the Heuristic Miner, it appears simpler to deal with a unique high grain trace than to indicate various precise but noisy traces.

The ETM execution time was extremely long (several hours on our target computer), and the computation failed several times due to memory lack (i.e. java heap space). Even when reducing by 50% the dataset content, the best fitness provided by the ETM was only of about 0.59 (with Naïve labelling). This result is consistent with Weerdt's observation (De Weerdt *et al.*, 2012) that the genetics approaches are not very suitable to process real life data. On the contrary, the Alpha Robust Miner provides no result when applying the Naïve strategy. It becomes more efficient when splitting the data into use cases. It provides results between 0.40 and 0.46. We attribute this to a better noise robustness.

Finally, the Fuzzy miner provides the best results (~73%) with the "1EGRC1XES" / "UHTD" case. However, the discovered model is unreadable and generates a dense "spaghetti-like" process. In addition, a manual analysis of some traces replays shows that many transitions noted as wrong in the model exist. Unlike the Heuristic Miner, the Fuzzy miner is better with precise tagging techniques.

Table 2. Summary of experiment results

Converter	Labelling	Nb Traces	Nb Events	Heuristics Miner Fitness (0-1)	Fuzzy Replay %	ETM Fitness 0-1	Alpha Robust Miner replay score
1EGRC1XES	Naive	1	227 782	0.4084	61.77	0.59	-
1EGRC2XES	Naive	1	455 564	0.5351	47.07	0.59	-
1EGRC1XES	UT	104	227 782	0.5554	57.54	-	0.44
1EGRC2XES	UT	104	455 564	0.4036	43.80	-	-
1EGRC1XES	UTD	6 617	227 782	-1.6116	56.76	-	0.40
1EGRC2XES	UTD	6 638	455 564	-0.3839	39.68	-	0.45
1EGRC1XES	UHTD	12 794	227 782	-0.2543	73.03	0.51	0.41
1EGRC2XES	UHTD	12 840	455 564	0.2346	56.16	-	0.46
1EGRC1XES	USHTD	8 882	141 828	-0.1706	68.78	-	0.52

5. Discussion and threats of validity

None of the tested techniques provides satisfying results with our data, even though we tested multiple conditions and provided ProM with logs of various granularity of events and several use case labelling strategies. We found that some of the methods have trouble to simply provide results. This could be an implementation problem or a lack of memory, but our tests highlight that with large datasets, some algorithms require very powerful machines to reach their objective without causing issues. In addition, we observed that getting reasonably good results does not mean that the resulting graph is readable. Our example with the Fuzzy miner is quite a good illustration of this limit. Our software experts indicated a high level of wrong transitions on the generated models. This brings us to a well-known problem: the Oracle. How confident in our results can we be? Moreover, 73% of replay indicates that we still have 27% of noise.

Furthermore, despite the application of an equivalent methodology and protocol, our results are in opposition with the results obtained by Astromskis (Astromskis *et al.*, 2015). The main difference between their study and ours lies in the implementation of the logging system and the size of our software. We believe that the automatic and generic implementation of our logging system could be the main explanation to the differences between our conclusions and Astromskis et al.'s. In fact, the logging system was manually added in their work. This means that the logging software was designed a posteriori for their purposes. On the opposite, in our case, the logging system was designed and implemented long time before our research. Nevertheless, the lack of information on the complexity of the analyzed software is not sufficient to explain reliably our failure. Some of the characteristics of our data might also be a reason. We detail them in the next section.

6. Learned Lessons

After obtaining the results we conducted a qualitative analysis to understand why process mining techniques deprived us from satisfying results. Our analysis led us to understand many specificities about our data which were directly induced by the nature of the user interface and the users' practices. We illustrate the problem in the next sub-section with the secretary scenario.

6.1. The Town Hall Agent Scenario

The "town hall agent scenario" is a typical work case which explains the nature and complexity of our data. The scenario starts with a town hall agent who starts a business scenario (BS1) thus creating events. A phone call occurs interrupting the case; BS1 is suspended. During the phone call, the agent starts a new scenario (BS2) to fulfill the caller's request. This triggers the creation of some other log events. The phone call ends but the agent still must perform actions related to BS2. At this point, someone enters the office and makes a new request. Again, this interruption suspends the processing of BS2. The agent starts a third scenario (BS3). When it ends, the secretary can resume BS2 and produce associated events. Finally, BS2 ends and BS1 is resumed.

This sequence of actions is possible because: 1) software users can be interrupted while executing a use case and 2) the software allows to start one or several new use cases simultaneously. This flexibility between use cases and tasks is mandatory to provide efficient tools to support this kind of activities. The understandability of software logs consequently suffers from this type of practice and thus our ability to understand the tasks.

6.2. A combination of four features

Overall, following the previously described scenario, we have identified that our data is characterized by four features which, all combined, make the business model hard to discover:

Unlabeled Use Cases. The first issue is related to the impossibility of labelling use cases on the fly. The users require flexible software that allow multiple use cases in parallel thus there is no clear indication in the log about when a use case starts and stops. This is a strong tendency in modern-web apps, such as Single Page Applications, enabling more flexibility, interactivity and overall a better user-experience to end-users (Mesbah & van Deursen, 2007).

The Impossibility of making Temporal or Structural Assumptions. Our data provide no clear indication about the duration of tasks. In fact, depending on the situation, filling a form to update a civil status can take from a few minutes up to several hours due to the interruptions and the requests. Moreover, some of the forms and screens are used by multiple use cases. For example, declaring a newborn may require updating the civil status of a parent. This parent civil status update will be the same as for a new passport request. The nature of our logs does not enable to identify clearly which specific screen is used in which use case.

The Presence of Loops. The nature of the users' activities also involves many redundant activities. For instance, the agent can come back and forth between the home menu and the burial plot attribution form. This enables the users to fill forms in a row in order, for instance, to process a large quantity of records. This, again, is quite common in administrative activities and will have to be identified when applying analysis techniques on our logs.

Multiple Levels of Interweaving. As illustrated in the scenario on Figure 6, we found multi-level of interweaving between use cases. The logs represent a list of events where each one contains a starting timestamp and ending one. These timestamps might overlap with other ones in the entire content of the log. This interweaving is due to the ability of users to concurrently start, perform and close several use cases. Our logs do not contain one linear story but several interweaved situations. It appears that existing approaches have difficulties handling the secretary scenario depicted in our case.

These four features are today more or less manageable by different approaches. For example, Walicki (Walicki & Ferreira, 2011) is known to be suitable for interweaving and Pourmirza approach works for cyclic applications (Pourmirza *et al.*, 2015). However, the combination of these four features makes the problem very hard to solve.

6.3. Challenges with logging systems

We also investigated the logging system itself. Our analysis highlighted interesting aspects of the logging system and difficulties to enhance it. In our case study, the logging system is implemented in a transversal manner, ensuring a strong decoupling between the application modules. This enables developers to rely on a generic tracing system and to focus on functional features development. The logging system uses the minimal data provided, thus the event logs are inaccurate. To make the logs more detailed, the entire logging system would have to be redesigned; whereas it is financially impossible. This problem is not restricted to our application. In general, since the logging system is not a business feature, it is not a major concern for the production team. The fewer resources are involved to set up a logging system, the better. The existing logging systems can produce events with technical details used for debugging purposes but are not able to inform the case id necessary for process mining. Multi-layer applications increase the difficulty because the information needed to create the correct event is spread over several layers. Several logging systems may be mandatory, or a single central logging system, that may break with a decoupled architecture.

Finally, redesigning the logging system of a large part of legacy code may cause regression. In fact, in some specific situations, adding logging calls can have unexpected impact on performances or even on the reliability of the execution. This problem needs great attention, since most of the practical use cases are unknown, non-regression tests are potentially inefficient. This analysis of our logging system stresses the tension between being able to build a generic and transversal mechanism and collecting enough data to uncover the business models afterwards. This type of

issues solely appears when working on large, distributed applications, where modifying the code manually is not a viable option. Overall, this highlights the need to work on logging systems architecture to enable genericity and low granularity of collected events.

7. Perspectives and Conclusion

In section 2, we have indicated the main reasons making it extremely difficult to understand our software logs. These difficulties amount to matching the events of a process with the software task model. This matching can be formalized by a multi-criteria optimization problem where each event must be associated to a unique task instance according to the log history, and each task must try to be fully described by events. This leads to a huge search space due to the combinatorial explosion. For this reason, we believe that Artificial Intelligence approaches might be of interest to process software logs. These approaches can preprocess the log to obtain labelled logs and, hence, prepare the logs to the application of process mining techniques. Two domains are interesting: 1) the clustering and classification approaches like the artificial neural networks and support vector machines, 2) the Multi-Agent Systems (MAS), in particular the Adaptive Multi-Agent Systems (AMAS).

The first approach tries to manage and reduce the complexity and the non-structuration of some processes. Rocío (Rocío *et al.*, 2015) have done a systematic literature review on Process Mining with artificial neuronal networks and vector machines approaches. Only two studies propose a contribution on process discovery and only Song's work explicitly uses the model and workflow defined by Process Mining (Song *et al.*, 2013).

On the contrary, the second approach does not try to reduce or simplify the complexity of data but tries to grasp the entire complexity of a problem. Some MAS rely on emergent techniques to find solutions. For example, Adaptive Multi-Agent Systems (AMAS) (Di Marzo Serugendo *et al.*, 2011) have already proved efficiency to solve some complex problems like big data analysis (Belghache *et al.*, 2016) even in very noisy environments. AMAS exploits the collective intelligence of agents to organize themselves and build a business process model.

To produce a suitable product while more complexity is needed, future software will have to dynamically adapt themselves to the users. To design these systems, we cannot rely on a known business process model. As the system cannot be intrusive, activities user traces must be taken as input. Regarding our needs, the process mining seems to be able to provide models of our software from execution logs. Because our events are unlabeled and our logging system unmodifiable, we used similar preprocessing steps as existing studies. Then, a set of process mining techniques was executed including the ProM plugins. The results showed an important quantity of wrong transitions in the discovered processes. This result highlights the complexity of our data: interweaving of use cases, lack of use case labels and various loops creating wrong trace labelling. Because of the diversity of

users and use cases, we could not make temporal, structural, behavioral or semantic assumptions. To deal with the complexity and removing the noise, we consider exploring artificial intelligence approaches and particularly multi-agent systems as a potential solution.

Bibliography

Ailenei I., Rozinat A., Eckert A., van der Aalst W. M. P. (2011). Definition and Validation of Process Mining Use Cases. In *Proceedings of Business Process Management Workshops*, pp. 75-86, Springer Berlin Heidelberg, Clermont-Ferrand, France.

Alves de Medeiros A. K. (2006). *Genetic Process Mining.* Thesis in Computer Science, Eindhoven University of Technology.

Astromskis S., Janes A., Mairegger M. (2015). A process mining approach to measure how users interact with software: an industrial case study. In *Proceedings of the International Conference on Software and System Process*, pp. 137-141, ACM Press, New York, USA.

Belghache E., George J.-P., Gleizes M.-P. (2016). Towards an Adaptive Multi-agent System for Dynamic Big Data Analytics. In *Proceedings of the Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*, pp. 753-758, IEEE, Toulouse, France.

Buijs J. C., van Dongen B. F., van der Aalst W. M. P. (2012). On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. In *Proceedings of On the Move to Meaningful Internet Systems*, pp. 305-322, Springer Berlin Heidelberg, Rome, Italy.

Cook J. E., Wolf A. L. (1998). Discovering models of software processes from event-based data. *Transactions on Software Engineering and Methodology*, vol. 7, no 3, pp. 215-249, ACM.

De Weerdt J., De Backer,M., Vanthienen J., Baesens B. (2012). A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. *Information Systems*, vol. 37, no 7, pp. 654-676, Elsevier.

Di Marzo Serugendo G., Gleizes M.-P., Karageorgos A. (2011). *Self-organising software : from natural to artificial adaptation.* Springer.

Günther C. W., van der Aalst W. M. (2007). Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *Proceedings of International conference on business process management*, pp. 328-343, Springer, Brisbane, Australia.

Lehman M. M. (1980). Programs, life cycles, and laws of software evolution, *Proceedings of the IEEE*, vol. 68, no. 9, pp. 1060-1076, IEEE.

Mesbah A., & van Deursen A. (2007). An Architectural Style for Ajax. In *Proceedings of Conference on Software Architecture*, pp. 9-9, IEEE, Mumbai, India.

Pérez-Castillo R., Weber B., Piattini M. (2013). Correlation of Business Activities Executed in Legacy Information Systems. In *Proceedings of Evaluation of Novel Approaches to Software Engineering*, pp. 48-63, Springer, Warsaw, Poland.

Pourmasoumi A., Bagheri E. (2017). Business process mining. *Encyclopedia with Semantic Computing and Robotic Intelligence*, vol. 1, no 1, World Scientific Publishing Company

Pourmirza S., Dijkman R., Grefen P. (2015). Correlation Mining: Mining Process Orchestrations Without Case Identifiers. In *Proceedings of Service-Oriented Computing*, pp. 237-252, Springer, Goa, India.

Rocío A., Maita C., Martins L. C., Ramón C., Paz L., Peres S. M., Fantinato M., Chopra A., Singh B. J. (2015). Process mining through artificial neural networks and support vector machines: A systematic literature review. *Business Process Management Journal*, vol. 21, no 6, pp. 1391-1415, Emerald Group Publishing Limited.

Salinesi C. (2017). Un jour, les Systèmes d'Information se concevront eux-mêmes. In *Proceedings of INFormatique des ORganisations et Systèmes d'Information et de Décision*, pp. 5-6, INFORSID, Toulouse, France

Song M., Yang H., Siadat S. H., Pechenizkiy M. (2013). A Comparative Study of Dimensionality Reduction Techniques to Enhance Trace Clustering Performances. *Expert Systems With Applications*, vol. 40, no. 9, pp. 3722-3737, Elsevier.

Spiliopoulou M., Pohle C., Faulstich L. C. (2000). Improving the Effectiveness of a Web Site with Web Usage Mining, In *Proceedings of Web Usage Analysis and User Profiling*, pp. 142-162, Springer, San Diego, CA, USA.

Srivastava J., Cooley R., Deshpande M., Tan P.-N. (2000). Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. *SIGKDD Explorations Newsletter*, vol. 1, no. 2, pp. 12-23, ACM.

Subramanyam R., Krishnan M. S. (2003). Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects. *Transactions on Software Engineering*, vol. 29, no. 4, pp. 297-310, IEEE.

Thompson D. V., Hamilton R. W., Rust R. T. (2005). Feature Fatigue: When Product Capabilities Become Too Much of a Good Thing. *Journal of Marketing Research*, vol. 42, no. 4, pp. 431-442, American Marketing Association.

van der Aalst W. M. P. (2016). *Data Science in Action*. Springer.

van der Aalst W. M. P. (1998). The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems and Computers*, vol. 8, no. 1, pp. 21-66, World Scientific Publishing Company.

van der Aalst W. M. P., Weijters T., Maruster L. (2004). Workflow Mining: Discovering Process Models from Event Log. *Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128-1142, IEEE.

Walicki M., Ferreira D. R. (2011). Sequence Partitioning for Process Mining with Unlabeled Event Logs. *Data & Knowledge Engineering*, vol. 70, no. 10, pp. 821-841, Elsevier.

Weijters A. J. M. M., van der Aalst W. M. P., Alves de Medeiros A. K. (2006). *Process Mining with the Heuristics Miner-Algorithm*. Technische Universiteit Eindhoven, Tech. Rep. WP, vol. 166, pp. 1-34.

Wen L., Wang J., Sun J. (2006). Detecting Implicit Dependencies Between Tasks from Event Logs. In *Proceedings of Frontiers of WWW Research and Development*, pp. 591-603, Springer, Harbin, China.