FERRET MINER: A Process Mining Case Study

Martin Alvarez-Lopez[1] Carlos Hernandez[2] Hardy Leung[3] Divyam Sobti[3]

***Abstract*—This project is a process mining case study designed to identify patterns and delays that obstruct the efficient flow of process activities. Specifically, our objective is to analyze event data logs from the travel reimbursement process at Eindhoven University of Technology (ETU/c), identify bottlenecks in the process, and seek improvement to its efficiency. We prepare and filter the event data, and then generate simplified and representative process models to locate the most time-consuming and impactful critical tasks. Our analysis suggests that improvement in the supervisor approval and payment handling activities could significantly reduce the time processing the travel reimbursement.**

***Index Terms*—process mining, data, conformance, event logs**

I. INTRODUCTION

Nowadays, most organizations use information systems to support the execution of their business processes, generating significant amounts of data. Mainstream information systems, such as Enterprise Resource Planning (ERP), Work on Management Systems (WMS), and Customer Relationship Management (CRM), have become vital tools behind the operation of almost any company process [15]. All of these platforms offer rich event logging capabilities, in which an event log is basically a record of an activity performed with a timestamp embedded [6].

There are plenty of oppportunities, through proper supervision of business behavior, to identify and adjust tasks that are generating problems for the business processes [9]. One important technique towards such goal is process mining, a specialized data analysis technique that reveals core business behavior indicated by activity logs. Process mining would start with these event logs, extracting important information including the chronology of the activities and their execution time registered in each log. Representative models would be built, or discovered automatically, and then be used to detect errors and deficiencies in the process, and to provide insight and suggestions for process enhancement. Process mining is most closely related to data mining amongst business intelligence disciplines. However, data mining concentrates on finding relationships among data, whereas process mining also looks at data from a systematic perspective.

A. Problem Description

In our process mining case study, we investigated a dataset with real-life event log data from the travel reimbursement process at Eindhoven University of Technology (ETU/e). Through analyzing the event logs and mining the business process underlying the events, we seek to identify and suggest remedy to address deficiencies in the reimbursement process. The event logs provide a detailed record of the travel reimbursement process, including important attributes such as identification, resource, timestamp, permits, payment and budgetting information, and event labels. Students or staff would submit their reimbursement application, provide suppport documentation (if traveling internationally), wait for reimbursement decision and subsequent payment, and possibly deal with rejection and resubmission. All of these information is captured in the event log. Collectively, these information provided a substantial amount of detail into the process. The data was obtained from the annual BPI Challenge as part of the 2020 International Conference on Process Mining [2].

Our approach to process mining is based on the work by Wil van der Aalst in his seminal paper titled simply "Process Mining" [16], where process mining guidelines are detailed. A process mining implementation generally covers three main issues: (1) discovery, (2) conformance, and (3) enhancement. Our project focuses on discovery since it is the area that requires more effort and it is the most deterministic for the success of the analysis. We will also take a cursory look at conformance as both an aide to discovery and future work.

B. Motivation

Publications on practical application of process mining are scarce, and our research paper aims to add support to this section. Our goal is to demonstrate the practicality of process mining analysis in practice, in application to the real-life travel reimbursement problem as detailed earlier.

Furthermore, we are very interested in the evaluation and application of a relatively new process mining technique based on PM4PY, a Python process mining library developed by Berti et al. [1]. According to the authors, PM4PY is a modern, open-source, and emerging platform developed specifically for the data science community. Prior to the turn of the millenium, there were hardly any tooling for process mining, but since then a slew of software tools, both open-source (PROM and APROMORE) and commercial (DISCO, CELONIS, and PROCESSGOLD), have been developed. Yet, many were proprietary, hard to extend, or hidden behind graphical user interfaces. The mindset of process mining tooling contrasted greatly with the modern philosophy of data science R&D, which put significant emphasis on openness and extensibility. PM4PY was developed to bridge the gap between process mining and modern data science capabilities including PANDAS, NUMPY, SCIKIT-LEARN. We whole-heartedly agree with the assessment of the authors, and we look further at opportunities that the

[1] MS Candidate in Software Engineering, San José State University
[2] MS Candidate in Computer Engineering, San José State University
[3] MS Candidate in Artificial Intelligence, San José State University

PM4PY approach brought forward, including the adoption of modern and rapidly evolving machine learning functionalities to process mining.

C. Approach Summary

The document is structured as follows: Sections I and II offer an introduction to process mining and a literature survey. Sections III, IV and V describe the technical details of our methodology, as well as the results of our investigation. We will end the paper with future work in Section VI and conclusion in VII.

II. Survey

In this section, we shall offer a very brief review of the literature on process mining, focusing on the development of techniques and algorithms from a control-flow discovery perspective, as well as its applications.

A. Process Mining

Van der Aalst et al. described the discipline of process mining as positioned somewhere between data mining and process modeling [17]. Process mining interacts with the rest of the software ecosystem mainly through the language of event logs (see Figure 1), and the objective is to provides useful information with implication on the business processes, organizations, and people involved.

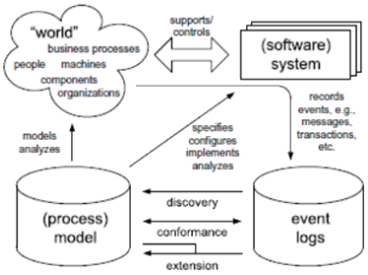

Fig. 1. General outline of process mining approach.

Although information system databases capture important business data in the form of event logs, they do not usually provide understanding of the data in a structural or systematic manner. Traditionally, process analysts must search for and retrieve the information directly from the system, and perform manual analysis to accomplish their specific objectives. To put it another way, event logs only possess the right information, but not the right insight. Whether or not a process mining implementation is successful or applicable thus largely hinges on what it can do with the event logs [16]. As such, process mining approach must embrace three main actions (Figure 2):

- *Discovery* – create robust and representative process models through analyzing event logs.
- *Conformance* – verify and compare newly created models against either the event logs or other predefined models to gain knowledge.
- *Enhancement* – turn the analysis into insights and suggestions to overcome previously detected problems.

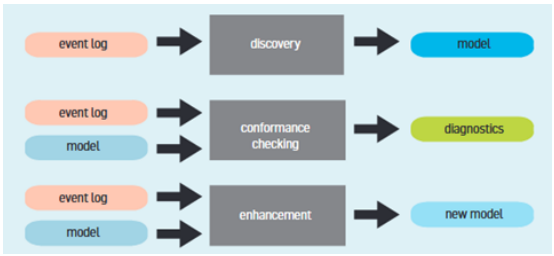

Fig. 2. General outline of process mining approach.

B. Process Models

Many distinct data analysis approaches have been developed to discover processes from event logs. However, PETRI NETs[1] have become a widely adopted technique over other methods [4] due to their intuitiveness, mathematical soundness, and ease of analysis. Another approach utilizes Unified Modeling Language (UML) diagrams [14]. In 2004, van der Wilst et al. described the a break-through algorithm in process mining called ALPHA MINER [17] which is capable of depicting event data logs into work-flow diagrams, which can be translated to PETRI NETs. The success of ALPHA MINER has inspired many follow-up publications, including a popular model called FLEXIBLE HEURISTIC MINER [18]. One main characteristic of this algorithm is its high level of configurability and flexibility. In order to deal with the massive number of events that actual information systems generate, Burattin et al. [3] developed another extension to HEURISTIC MINER based on lossy counting, as well as a sliding window that effectively analyzes streams of real datasets.

C. Applications

Most process mining publications are dedicated mainly to the enhancement of novel methodologies and algorithms with significant emphasis on the control-flow discovery area [17]. However, the public services area has been fruitful ground for process mining deployment research. For instance, a genetic miner approach was tested by de Madeiros et al. [5] in Dutch municipalities. Additionally, van del Aalst et al. [12] applied an analysis of organizations in another municipality process. Rozinat et al. [13] evaluated process models in two different case studies on the public sector. Nonetheless, all these research applications were focused on validation and assessment, rather than to improve the process.

Implementation of process mining approaches has also been done in the private domain. Distinct discovery approaches were deployed by Goedertier et al. [7] in the field of telecommunication. Mans et al. [10] and Rebuge et al. [11] both evaluated process mining methods to resolve the tracking of patients within the healthcare industry.

III. Technical Approach

We shall now briefly describe our approach to finding where the process bottlenecks are. We would start with a thorough

[1] and the equivalent BPMN models and process trees

analysis and cleaning of the event logs, and then build models that best capture the behavior of the process, making the proper tradeoff between robustness (does the model capture the intent of the process), and coverage (what fraction of the use cases are captured). Then, we will use the chosen models to answer the questions we have in mind.

We shall note that process mining is a relatively new techniques, and as such we spent a disproportional amount of time ramping up the project. While we fully believe in the full potential of PM4PY due to its focus on algorithmic discovery and availability on Python, it is essentially still an academic software in its nascent stage, and hence suffers from instability. There is still plenty of room for improvement, such as a lack of control in visualization and presentation, scanty documentation, and signs of immaturity in API design.

IV. Methodology

In this section we will discuss about the data, how we preprocessed it, what model we used, and how we reached the results and the insight we obtained.

A. Description of the Dataset

After a thorough review of what our tasks entail, we decided to focus exclusively on the domestic and international travel event logs in order to identify the process bottlenecks. Both event logs were provided to us in XES format, an XML-based standard for event logs [8], which can be easily parsed using PM4PY. The events are organized as a two-level hierarchy as follows: An event log is made of a list of cases (also known as traces in technical term), and each case is made of a list of events. Cases and events are both extensible constructs to allow for maximum flexibility in event modeling.

The event logs provide a detailed record of the travel reimbursement process, including important attributes such as identification, resource, timestamp, permits, payment and budgeting information, and event labels. Students or staff would submit their reimbursement application, provide suppport documentation (if traveling internationally), wait for approval and payment, and resubmit a corrected request if rejected.

As the logs were imported into Python via PM4PY, they are represented as dataframes, a key construct of PANDAS, the popular Python Data Analysis Library. However, as dataframes are not hierarchical, PM4PY flattene the log so that the enclosing case and the case attributes become attributes of an event. If there were A_c attributes associated with a case, and A_e event attributes in the XES specification, after PM4PY import, we would obtain a PANDAS dataframe with N events, each with a total of $A_c + A_e$ attributes.

In our project, the domestic process log is organized differently than the international process log. For domestic trips, employees would first complete the trip and then ask for reimbursement. On the other hand, the policy for international trips mandates that employees first obtain travel permits from the supervisor prior to commencing the trips.

As a result, we are dealing with a more complex process in the case of international travel, and the formal specification that describes international cases also contain more fields. Specifically, domestic cases have 5 attributes each, and international cases have additional 13 attributes, together with the 5 attributes shared with domestic cases (`id`, `concept:name`, `BudgetNumber`, `DeclarationNumber`, `Amount`), for a total of 18 attributes. Regardless of the type of trip, the event in the XES hierarchy has the same 5 attributes (`id`, `org:resource`, `concept:name`, `time:timestamp`, and `org:role`). After flattening, PM4PY returned a domestic dataframe with 56437 events and $5 + 5 = 10$ attributes, and an international dataframe with 72151 events and $18 + 5 = 23$ attributes. To avoid namespace collision, all case attributes were prefixed with `case:` by PM4PY (for example, the `id` field of a case became `case:id`). Please refer to Table I for the complete list of attributes:

TABLE I
XES Event Attributes (5 Total)

XES Event Attributes (5 Total)
id, org:resource, concept:name, time:timestamp, org:role
XES Domestic Case Attributes (5 Total)
id, concept:name, BudgetNumber, DeclarationNumber, Amount
XES International Cases (18 fields)
Permit travel permit number, DeclarationNumber, Amount, RequestedAmount, Permit TaskNumber, Permit BudgetNumber, OriginalAmount, Permit ProjectNumber, concept:name, Permit OrganizationalEntity, travel permit number, Permit RequestedBudget, id, Permit ID, Permit id, BudgetNumber, Permit ActivityNumber, AdjustedAmount

Prior to process mining, we performed a thorough cleaning of the domestic and international dataframes to ensure we have the best possible starting point for analysis. There were three categories of modification we applied to the datasets:

B. Data Preprocessing I – Missing Values

While we did not find any missing value per se (as far as what PANDAS is concerned), we did find columns with string values UNKNOWN or MISSING. We carefully evaluated what to do with these attributes on a case-by-case basis. For example, we could drop the column (attribute), drop the row (event), fill the missing values, or simply ignore the column.

In the case of international reimbursement, we decided that we should drop events without a permit number. The permit number is actually a concept of the case, and hence we were removing traces without permit number in their entirety. This would turn out to be the most significant modification to our international dataset. We believe this is justified, because a trip without any associated permit violated the very fundamental assumption about the international reimbursement process, that there is a permit associated with the trip. Note that the decision to drop these entries was not "lossy", because international reimbursement request without a permit number *should* have been rejected and not entered into the system in the first place.

C. Data Preprocessing II – Duplicated Attributes

Perhaps due to prior inconsistency between versions of the enterprise software, there are fields that are similar or duplicate of each other. For example, `Permit ID`, `Permit id`, and `Permit travel permit number` are one and the same. Moreover, it is quite likely that exactly one of the three fields were filled, while the others are missing (or reported as MISSING. We combined these similar fields into one, picking out the value from the non-missing column, and dropped the duplicates. Note that we performed this duplicate analysis prior to future treatment of missing values.

As a result of these cleaning procedures, the size of the the international dataframe was reduced from 72151 to 43707 (a 60% drop), with the missing permit accounting for 95% of such reduction. The number of columns was reduced from 23 to 19, since we dropped several permit-related columns that are duplicates. Similarly, the domestic dataframe was also modified to a much small extent. The number of events was reduced from 55087, a mere 2% drop, and the number of columns stayed the same.

D. Data Preprocessing III – Infrequent Events

Process models that can explain all possible event traces could be arbitrarily complex. For example, obviously we would expect reimbursement to be paid out only after the request was approved. But what if there is a trace with a missing approval due to erroneous logging? A process model built to handle 100% of traces must handle a flow with payment before request, however bizzare such trace is. Not only would the model be more complex and unwieldy, it would also be incorrect and fail our objective.

A naive approach would be to remove events that are uncommon, but doing so may not be sufficient. Consider the previous example; while REQUEST APPROVED that happened after PAYMENT HANDLED might be a rare sequence, neither of these events were rare.

In PM4PY terminology, the variants of an event log is the union of all possible sequences of event execution (ignoring all but the event labels). For example, there were 68 variants in the domestic log, and 464 variants in the international log. In fact, the above example points to the need to remove traces that are uncommon. Clearly, it could be beneficial to have the choice to build the process model based on only the most popular traces. Indeed, this was an important part of our methodology and we shall take a closer look in the next section.

V. Models and Results

With the event logs well understood and properly prepared, we are now ready to build the actual process model.

A. Choice of Model

We must first answer the question of what formal model to use. Our choices include BPMN[2] model, process tree, PETRI NET, and Directly-Follows Graph (DFG). The first three are mathematically equivalent and inter-changeable. On the other hand, DFG shows a different view of BPMN graph constructed by aggregrating all possible link between any pair of events that follows each other.

There are pros and cons in choosing PETRI NET vs DFG. DFGs can be constructed directly from the event logs, and are simple to understand. PETRI NETs (as well as the equivalents) can only be generated heuristically, for example, through inductive or heuristics miners (which PM4PY supports). The difference mainly lies on how concurrency is handled. For example, say there are two activities X and Y that are sandwiched between A and B. In other words, the following are both legitimate sequence of executions:

$$A \to X \to Y \to B$$
$$A \to Y \to X \to B$$

While both ALPHA MINER and INDUCTIVE MINER would correctly deduct (probabilistically) that X and Y have no inter-dependency but are instead concurrent events, DFG would not be able to discern that, because all it saw was Y directly follows X in the first example, and X directly follows Y in the second. Thus, concurrency would show up in DFGs as tight loops between two activities, yet there is no way to prove that tight loops imply concurrency. If indeed DFGs become too complicated, they will confuse rather than aide reasoning.

For our purpose, however, we still believe is the preferred approach as the statistics on the edges of a DFG can be made readily available, and more easily reasoned with. We just need to make sure we have a model that is not too complicated. This speaks to the importance of limiting the model size, which we shall discuss next.

B. Choice of Model Size

How many top variants do we use to build the model? We did experiment with different values of k, where k is the number of top variants to be used to build the DFG. We evaluated many factors, though more qualitatively than otherwise, and came to the conclusion that we would achieve the best balance between coverage and robustness if we keep approximately 15 to 25% of the top variants, depending on data (higher for domestic, and lower for international). Figures 3 and 4 shows the DFG models we picked for domestic and international reimbursement respectively. In each figure, the DFG to the left is annotated with the number of directly following edges, and the DFG to the right is annotated with the mean duration between the two events. Both models captured the vast majority of the important interactions very well and were reasonably easy to comprehend.

C. Results

Now that we have a robust process model built on reasonably clean data, we can now answer the question: Where are the bottlenecks in the process of a travel declaration?

The biggest bottleneck is the supervisor activities for both international and domestic declarations. For the supervisor to do the final approval and request payment, it takes an average

[2]BPMN stands for Business Process Model Notation

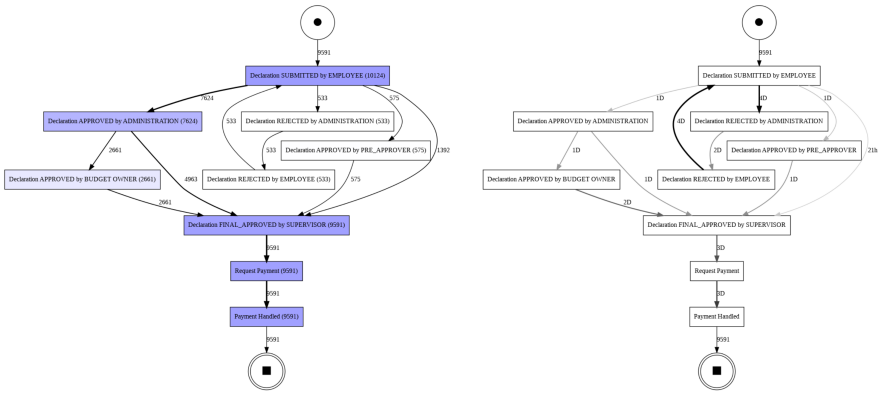

Fig. 3. DFG Model for Domestic Reimbursement.

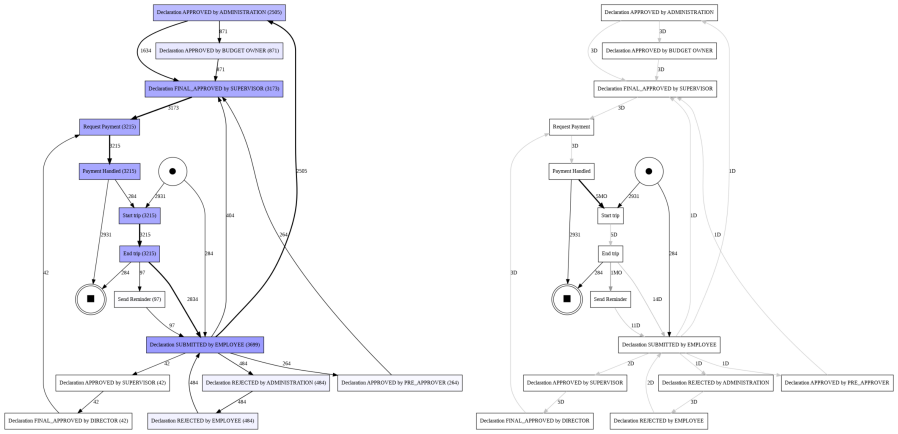

Fig. 4. DFG Model for International Reimbursement.

of 4 to 5 days. To find such bottleneck, we would identify that links with both high frequency *and* long duration. Since the models are simple enough, we were able to visually inspect the models and identify such link. We did add code to work around PM4PY and automatically identify such links.

On the other hand, if we were to identify one activity, PAYMENT HANDLED is the biggest bottleneck. That said, it might be outside of the organization's control and may be more of a financial institution issue.

VI. FUTURE WORK

Now we shall look into two potential area of work in the future – conformance analysis and machine learning.

A. Conformance Analysis

Once we have developed a process model either extracted through process discovery, or built on some reference guidelines. It would be interesting to investigate further into conformance checking to see how well the data fits the model, while in turn would help us evaluate and improve on the model.

This can be done with a method called token-based replay (TBR) on the PETRI NET converted from the process model. For each trace, we would try to see if it can be simulated on the PETRI NET which can be done precisely in a mathematically sound way. We define the total fitness simply as the percentage of traces that fit the model. Other techniques we can apply are alignment analysis and behavior analysis, though they both come at the expense of higher runtime complexity. Together these techniques can be used to better identify the hows and whys of non-conformance, and to build better models. In conformance analysis, tetting to 100% total fitness is not a desired goal, as explained earlier. Instead, what we seek is a tradeoff. On the one hand, you want the model to be robust enough to elegantly explain the process. On the other hand, we do not want to over-build a model so complex that it is impossible to reason about.

Let's use our specific problem as an example. We ran an experiment to examine the tradeoff between fitness and model complexity. We built the model multiple times with INDUCTIVE MINER, each time on a different value of k, where k is the number of top variants we used to build the model, and we recorded the total fitness via conformance check, as well as an estimate the complexity of the model computed using custom Python code. In Figure 5, the red line is the total fitness, and the blue line is the size of the model. We see that, as we include more top variants, the model gets more complex, but the fitness also improves. If we would include

only the top 3 variants, we can already explain 90% of the traces. But if we blindly crank up the knob to 18 variants, the model would now account for 99% of the traces, but the it is 5 times bigger, and much harder to understand.

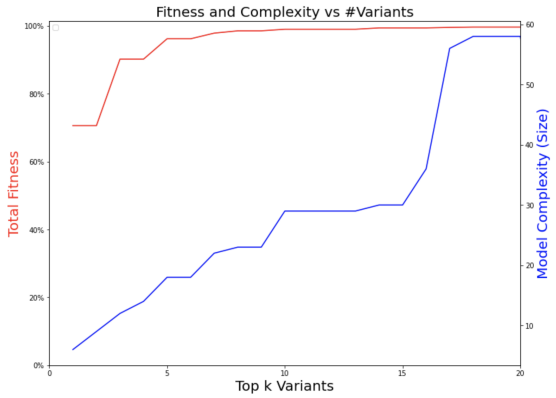

Fig. 5. Fitness and Complexity vs Top k Variants.

The sweat spots happen at $k = 5$, which can explain 96% of the traces. Moreover, it was able to properly capture the submission, approval, rejection, resubmission, and the final payment. It represents the best tradeoff between fitness and model complexity. It would be interesting to further expand the idea of conformance checking as a way to help build the model, and we have only just scratched the surface.

B. Machine Learning and Timely Analysis

Another area of interest is to apply machine learning techniques to perform (1) flow analysis to predict likelihood of resubmission, rejection, payment delay, or over-budget based on other variables (time since submission, request amount, prior owner approval), and (2) timing analysis to determine the average expected time before the trace completes. This provides important clues to administrators and employees alike about the time frame of payment in the best-case scenario. We are also interested in detecting non-compliance *as it happens* and sending alerts to administrators.

VII. Conclusion

In conclusion, we have completed a process mining case study and demonstrated the effectiveness of modern process mining techniques in identifying bottlenecks in the travel reimbursement process. We are encouraged by the power of the PM4PY platform and expect much more research and development to go into process mining, especially in the direction of process discovery in the era of machine learning.

References

[1] Alessandro Berti, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. Process mining for python (pm4py): Bridging the gap between process- and data science. *CoRR*, abs/1905.06169, 2019.

[2] BPI Challenge 2020. https://icpmconference.org/2020/bpi-challenge/. Accessed: 2022-11-11.

[3] Andrea Burattin, Alessandro Sperduti, and Wil MP van der Aalst. Control-Flow Discovery from Event Streams. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2420–2427. IEEE, 2014.

[4] Jonathan E Cook and Alexander L Wolf. Discovering models of Software Processes from Event-based Data. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 7(3):215–249, 1998.

[5] Ana Karla A de Medeiros, Anton JMM Weijters, and Wil MP van der Aalst. Genetic process mining: an experimental evaluation. *Data mining and knowledge discovery*, 14(2):245–304, 2007.

[6] Event Log – The Process Mining Glossary — Appian. https://appian.com/process-mining/event-log.html. Accessed: 2022-11-11.

[7] Stijn Goedertier, Jochen De Weerdt, David Martens, Jan Vanthienen, and Bart Baesens. Process discovery in event logs: an application in the telecom industry. *Applied Soft Computing*, 11(2):1697–1710, 2011.

[8] IEEE Standard for EXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. https://xes-standard.org/. Accessed: 2022-11-11.

[9] Frank Leymann and Wolfgang Altenhuber. Managing business processes as an information resource. *IBM systems journal*, 33(2):326–348, 1994.

[10] Ronny S Mans, MH Schonenberg, Minseok Song, Wil MP van der Aalst, and Piet JM Bakker. Application of process mining in healthcare–a case study in a dutch hospital. In *International joint conference on biomedical engineering systems and technologies*, pages 425–438. Springer, 2008.

[11] Álvaro Rebuge and Diogo R Ferreira. Business process analysis in healthcare environments: A methodology based on process mining. *Information systems*, 37(2):99–116, 2012.

[12] A. Rozinat, I. S. M. de Jong, C. W. GÜnther, and W. M. P. van der Aalst. Process mining applied to the test process of wafer scanners in asml. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(4):474–479, 2009.

[13] Anne Rozinat, Ronny S Mans, Minseok Song, and Wil MP van der Aalst. Discovering simulation models. *Information systems*, 34(3):305–327, 2009.

[14] Shinobu Saito. Understanding Key Business Processes for Business Process Outsourcing Transition. In *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, pages 35–39. IEEE, 2019.

[15] Tutorial:Introduction — ProM Tools. https://www.promtools.org/doku.php?id=tutorial:introduction. Accessed: 2022-11-11.

[16] Wil Van Der Aalst. Process Mining. *Communications of the ACM*, 55(8):76–83, 2012.

[17] Wil Van der Aalst, Ton Weijters, and Laura Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE transactions on knowledge and data engineering*, 16(9):1128–1142, 2004.

[18] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process Mining with the Heuristics Miner-Algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166(July 2017):1–34, 2006.

VIII. Contributions

We would like to thank Professor Mahima Agumbe Suresh for her teaching, encouragement, and devotion to helping us and other students succeed. The following is a breakdown of our individual contributions:

- **Martin Alvarez-Lopez** – advocated for the original proposal, and set the overall direction of the project. He contributed to data preparation, the main body of work on process mining, prepared and gave the main part of the presentation.
- **Carlos Hernandez** – wrote the first draft of the term paper, completed with extensive literature survey, project outline, report template, as well as a significant portion of the writeup.
- **Hardy Leung** – did an early investigation on PM4PY, and worked on conformance analysis. He also migrated the paper to the IEEE template, and wrote the second half of the term paper.
- **Divyam Sobti** – worked closely with Martin A. on data preparation and conducted the main investigation on process mining. He set up the CoLab environment, and contributed to report writing as well.