ProcessExplorer: An Interactive Visual Recommendation System for Process Mining

Alexander Seeliger, Timo Nolle, Max Mühlhäuser
Technische Universität Darmstadt
Darmstadt, Germany
[seeliger,nolle,max]@tk.tu-darmstadt.de

ABSTRACT

More and more business process data is collected by organizations to analyze and optimize their process performance. As a consequence it is particularly challenging to locate possible process issues or potential optimizations using process mining. Process mining aims at analyzing the actual usage of information systems by reconstructing a process model from recorded event log. However, such large amount of data often leads to spaghetti-like visualizations which are in-comprehensive and inaccurate. This paper addresses this issue by introducing an unsupervised visual recommender system for process analysis. The system provides suggestions during the interactive visual inspection of the discovered process model by recommending points of interests (e.g., long duration times or exceptional process behavior) ranked by severity. For calculated interest points we characterize the deviation from the average behavior as well as compute the effect the observed conspicuousness has. Our approach has been implemented as a ProM plugin. We evaluate our approach by presenting a case study using a real life event log.

KEYWORDS

Process mining, Exploratory analysis, Process analysis, Visualization recommendation, Business process intelligence

ACM Reference Format:
Alexander Seeliger, Timo Nolle, Max Mühlhäuser. 2018. ProcessExplorer: An Interactive Visual Recommendation System for Process Mining. In *Proceedings of KDD 2018 Workshop on Interactive Data Exploration and Analytics (IDEA'18) (IDEA @ KDD'18)*. ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

Process mining gives organizations a deep insight into their business processes by extracting knowledge from event logs recorded by process-aware information systems (PAISs) [18]. The most used method in process mining is the discovery of process models from such event logs. In particular, organizations can actually see how specific processes are executed, identify potential bottlenecks, detect compliance violations and other suspicious parts of the process. Besides academic tools like ProM, a wide range of commercial process mining discovery tools exists. Using such tools, one can visually inspect the actual process model to extract such valuable knowledge, like identifying customers with exceptional high work effort or slow process activities. However, the exploratory visual inspection of the model becomes more and more challenging due to the increase of recorded business data as well as the increase of process complexity. The discovered process model from real data often leads to in-comprehensive and imprecise process models, also called spaghetti models. In such cases, the manual filtering of the dataset and exploratory inspection of the model is hard, and finding the interesting parts of the process in a spaghetti-like model is time-consuming and error-prone.

To improve the exploration of process models and corresponding attributes, visualization tools might provide certain automation to generate a set of interesting visualizations. Interesting in this case means to show the user subsets of data that are significantly different to the overall dataset. However, for high dimensional datasets, the choice for appropriate attributes, data transformations and visualizations is very high, so exploring all potential combinations is not possible. While certain approaches for data tables exists in the data mining community, appropriate filtering and recommendation methods are also required in the area of process mining.

In this paper, we present *ProcessExplorer*, an interactive process mining system that recommends process analysts interesting data subsets and visualizations to support the exploration of high-dimensional process data. Our system respects the four process perspectives (functional, control-flow, organizational and data) and can be easily extended (e.g., process performance indicators). ProcessExplorer takes an event log as the input and automatically analyzes the dataset to provide interesting visualization recommendations. To overcome the amount of data, the user is shown a ranked list of interesting data subsets that can be interactively explored. The ranking of the findings is based on a custom interestingness measure. The system shows the corresponding process model, which reflects the behavioral aspects of the selected subset, and additional data visualizations (insights) created from data attributes attached to the event log. Such visualizations are selected by deviation of means to focus only on the most relevant and interesting visualizations. We also show deviations between identified subsets of data, characterizing each data subset such that process analysts can easily see the differences of the subset recommendations.

In summary, the contributions of this paper are as follows:

(1) We present ProcessExplorer, an automated system that provides interesting visualization and data subset recommendations to support the exploratory analysis of event logs.
(2) We describe our visualization recommendation engine that takes an event log as the input to provide useful recommendations.

(3) We show the results of a case study to show the application of our recommendation engine to a real life event log and describe an usage scenario.

The paper is structured as follows. In section 2 we give a short introduction into process mining and a running example. Section 3 presents our approach. Section 4 shows the interactive browsing and exploration. Next, in section 5 we present our case study. In section 6 we introduce related work. Finally, we conclude and give some directions for future work.

2 PROCESS MINING

In process mining, the starting point of the analysis is an event log L. An event log stores information about the actual use of an information system, in particular, it stores which events have been executed at a certain time. The sequence of events, also called trace, is the life-cycle of a particular case (i.e., a process instance). Often an event log may also store additional information (e.g., the resource that executed a specific event) about cases and events. Based on the definitions in [18], we define the following notations:

Definition 2.1. (Event, Attribute) We define $\mathcal{E}$ to be the set of all possible event identifiers. Events may be described by attributes, such as the timestamp. Let $\mathcal{A}$ be the set of all attributes and $\mathcal{V}_a$ the set of all possible values of the attribute $a \in \mathcal{A}$. For an event $e \in \mathcal{E}$ and an attribute $a \in \mathcal{A}$, we define $\#_a(e)$ to be the value of attribute a for event e.

Definition 2.2. (Case, Trace, Event Log) We define C to be the set of all possible case identifiers. Cases can also have attributes, so for each case $c \in C$ and an attribute $a \in \mathcal{A}$, we define $\#_a(c)$ as the value of an attribute a for case c. A mandatory case attribute $trace$ is defined for each case: $\#_{trace}(c) \in \mathcal{E}^*$, also denoted as $\hat{c} = \#_{trace}(c)$.
A trace is a finite sequence of events $\sigma \in \mathcal{E}^*$.
An event log is a set of cases $L \subseteq C$.

Table 1 shows an excerpt of an example event log of a procurement process. It consists of several activities such as *Purchase request created*, *Purchase request approved*, *Purchase order created*, *Goods receipt*, *Invoice receipt* and *Payment*. The example event log consists of two cases $L = \{1, 2\}$, six events $E = \{11, 12, 13, 14, 21, 22\}$ and the attributes $\mathcal{A} = \{\text{Case id}, \text{Event id}, \text{Vendor}, \text{Category}, \text{Timestamp}, \text{Activity}, \text{Resource}\}$. The table shows the values of the attribute of each case, for example $\#_{\text{Resource}}(11) = \text{John}$.

3 METHOD

In this section, we introduce our subset and visualization recommender system. ProcessExplorer consists of three steps. First, the system searches for similar process behaviors, and clusters such behaviors together using trace clustering. Second, based on the selected process perspectives, we determine interesting visualizations by calculating the difference of means. Lastly, we define an interestingness measure that ranks identified visualizations and shows most interesting visualizations on the top of the user interface. The details of the steps will be presented in the following sections.

3.1 Extract Process Behaviors using Trace Clustering

The starting point of our method is an event log L, containing traces of ordered events and attributes. Usually, event logs contain more than a single process behavior which is often the reason for spaghetti-like process models, highly complex and complicated to understand. We perform trace clustering to segment the event log into smaller sublogs to reveal the different process behaviors, allowing the user to better understand the overall process.

Based on related work (see section 6), we define a custom distance function [16] which incorporates the control-flow and the data perspective. In particular, we first extract frequent itemsets using the FPClose algorithm [8] from the attribute values $\#_a(c)$ for all $c \in L$ and $a \in \mathcal{A}$ to reduce the search space for clusters. Itemsets are built by all cases following the same event sequence. The idea is that the control-flow behavior also highly depends on the data attributes. For example, in the procurement process, certain supplies exist that require additional approval activities, indirectly changing the behavior of the process. Extracted itemsets are compared and used to build candidate clusters. $dist_{itemset}(a, b)$ determines the itemsets in both cases and returns the proportion of agreement.

Furthermore, we use the Levenshtein distance metric over the event sequences to compare the deviation on the control-flow perspective. Both perspectives are merged into a single distance function which is then the input for agglomerative hierarchical clustering with ward-linkage.

Definition 3.1. (Combined distance function) We define a custom distance function which compares the frequent itemsets and the event sequence of two cases $a, b \in L$.

$$dist(a, b) = w \cdot dist_{itemset}(a, b) + (1 - w) \cdot levenshtein(a, b)$$

The optimal number of clusters is determined by maximizing the weighted ICS-fitness of the underlying process models and optimizing the clustering parameters (weighting factor w, number of clusters n and the minimum support θ of itemsets). The fitness is a measure which describes how much behavior of the event log is captured in the discovered process model. We optimize our clustering parameters using Particle Swarm Optimization with 5 particles and 10 iterations.

Definition 3.2. (Weighted ICS Fitness) Let ics_k the ICS-Fitness of a model k, then the weighted ICS Fitness is defined as:

$$ICS - Fitness = \frac{\sum_{k=1}^{N}(n_k \cdot ics_k)}{|L|}$$

As a result, we retrieve multiple sublogs which can be visualized separately.

Definition 3.3. (Trace Clustering) The clustering of event logs is a function $cluster : L \longrightarrow L^n$ that generates n sublogs from one event log with $L_i \subseteq L$ with $0 \leq i \leq n$, $n, i \in \mathbb{N}$ using the distance function $dist(a, b)$.

Our prior work [16] showed that trace clustering overcomes spaghetti models, and reveals hidden process behavior, not fully visible in an overall model. Trace clustering itself only returns possible subset selections, containing process behavior that might be

Table 1: Simplified example event log of a procurement process.

Case id	Event Id	Vendor	Category	Timestamp	Activity	Resource
1	11	B. Trug	Office supplies	2017-04-17 10:11	Purchase request created	John
1	12			2017-04-18 14:55	Purchase request approved	Maria
1	13			2017-04-18 17:12	Purchase order created	Roy
1	14			2017-04-29 09:06	Goods receipt	Ryan
2	21	Company	Computer	2017-04-19 17:45	Purchase order created	Emily
2	22			...	...	...

interesting to the user. However, it does not provide any explanations about the generated clusters, nor does it show the differences between clusters. Depending on the event log and the available data attributes, the number of clusters may be high such that users need to look at all clusters to find the interesting ones.

3.2 Defining an Interestingness Measure

After extracting the different process behaviors, we calculate the interestingness of each cluster, providing a ranking that shows the user which clusters are worth looking at. We define interestingness as the difference of means between the process instances in a cluster and a reference set. Analogue to SEEDB [20], the reference set (L_R) can be the entire event log (L), the complement of the cluster i ($L - L_i$) or a custom selection of instances (L'). For instance, clusters of cases might be interesting if the distribution of values of a specific attribute (e.g., number of involved resources, suppliers or event transitions) are different compared to the reference selection.

While Voyager [26] and SEEDB [20] inspect data tables, which compares to the data perspective in process mining, and process mining related work only focuses on the control flow perspective, our approach considers both process perspectives together. In particular, it is highly arguable that analyzing the control-flow perspective alone will provide insights for better processes [19]. The goal of BPM, and process mining, is to improve the business process rather than the underlying process model. Therefore, our approach is flexible enough to consider cost, quality, flexibility, time and KPIs for calculating the interestingness of clusters.

The total interestingness of a single cluster is determined by the combination of single measures and their effect on the cluster building. We distinguish between two types of interestingness measures:

- **Trace-based measure**. A trace-based measure calculates a value from a single trace that can be used to compare two traces with each other, such as the total duration time or the occurrence of resources.
- **Cluster-based measure**. A cluster-based measure calculates a measurement value from a set of traces in a cluster that can be used to compare two clusters. For example, one might want to compare the distribution of attribute values such as the vendor.

The choice which measures to select highly depends on the analysis goal and the interest of the analyst. In ProcessExplorer we include various different trace- and cluster-based measures which can be used (see Table 2). These measures were derived and selected from the process mining domain based on their use in existing tools. However, we would like to note that our method is not limited to the use of the presented measures, but can also be extended with custom or external measures like customer satisfaction.

Table 2: Trace- and cluster-based measures in the current implementation.

Measure		
Trace	**Control-flow**	Cost per event
		Cost per trace
		Loops
		(Directly) followed by
	Resource	Number of resources
		Resources involved
	Data	Attribute value
	Time	Duration of events
		Duration of trace
	Function	Occurence of events
		Co-occurrence of events
		Number of events
Cluster	**Resource**	Attribute resources
	Data	Distribution of attribute values

In order to avoid the consideration of irrelevant measures, we perform *statistical significance* tests. For each cluster we calculate if the distribution of the observed measure values in the cluster is different to the instances in the reference selection $\mathcal{L}_R$. To compare the distributions of trace-based measures we use the non-parametric Kolmogorov-Smirnov test evaluating the null hypothesis that the two samples come from the same underlying distribution. For cluster measures (e.g., data attribute value distribution), we calculate the difference by measuring the distance between the two distributions. We use the Kullback-Leibler Divergence to measure the deviations between the selection and the reference event log.

Definition 3.4. (Kullback-Leibler Divergence [12]) The divergence between two probability distributions P and Q is defined as:

$$D_{KL}(P \parallel Q) = -\sum_i P(i) \log \frac{Q(i)}{P(i)}$$

With the statistical significance test we gather if there is a significant measure difference between the instances within the cluster and the instances in other clusters. These tests do not make any conclusion about how large the difference of the values is. Even small differences between the values may lead to high significance results. To quantify the difference we calculate the effect size using *Cohen's d*.

Definition 3.5. (Cohen's d [3]) The effect size d returns how large or small the difference of the observed measure is. Cohen considers $0.2 < d \leq 0.5$ as a small effect, $0.5 < d \leq 0.8$ as a medium effect and $d > 0.8$ as a large effect. It is defined as follows:

$$\text{cohensd}(X_1, X_2) = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{(s_1^2 + s_2^2)/2}} \quad \text{with} \quad s_i^2 = \frac{1}{n_i - 1}\sum_{j=1}^{n_i}(x_{j,i} - \bar{x}_i)^2$$

For the calculating the overall interestingness of a cluster we only consider measures that are statistical significant and for which the effect size is at least 0.2 or a distribution difference of 0.25. The clusters are finally ranked by the average interestingness multiplied by the number of traces in the cluster.

3.3 Deviations and Data Explanation

The last step of our approach is to determine the deviations between clusters, and to highlight the most prominent measurements within a cluster. We use the same measures calculated for the cluster interestingness to extract the deviations and to explain the clusters. These explanations help to increase trust in the system because it directly shows the identified findings, explaining why a specific cluster is marked as interesting. For example, the system might found that the process cases in a certain cluster have a higher duration time compared to the duration in the whole event log. Such insights would normally be hidden, except the analyst would manually search for them.

However, we found that showing all interesting measurements would unnecessarily include duplicates or measurements that depend on each other. For example, the *co-occurrence of events* and the *followed-by* measurements are highly related to each other but showing both would not necessarily provide more knowledge. In order to overcome this issue, we cluster measurements together that have a high correlation between each other. We calculate the pairwise correlation between measurements using the Pearson correlation and use hierarchical clustering to group measurements together. The optimal number of clusters is determined by optimizing the silhouette coefficient.

For ranking the explanations, we use the uniqueness and extremeness of the explanation over all clusters. Explanations that only occur for a single cluster are much more interesting than explanations that occur multiple times for different clusters. So we count the number of occurrences and give unique explanations a higher rank.

Definition 3.6. (Uniqueness) The uniqueness of an insight i in a cluster $c \in C$ is defined as:

$$uniqueness(i, c) = 1 - \frac{\sum_{c_i \neq c \wedge i \in insights(c_i)}^{c_i \in C} |c_i|}{\sum_{c_i \neq c}^{c_i \in C} |c_i|}$$

$|c|$ is a shorthand for counting the instances in c, and $insights(c)$ is a function taht returns a set of insights discovered in cluster c.

As a second ranking score, we include the extremeness of the difference between the selected and the reference event log. The higher the difference, the higher the ranking.

4 PROCESSEXPLORER USER INTERFACE

Figure 1 shows the overall user interface of ProcessExplorer. On the lower bottom (4) the analyst can adjust the perspectives that he is currently interested in. By selecting one or more of the perspectives, different trace- and cluster-based measures are used to calculate the interestingness score. The analyst may also change the significance level and select the reference event log, used to calculate the deviation of means for the interestingness measure.

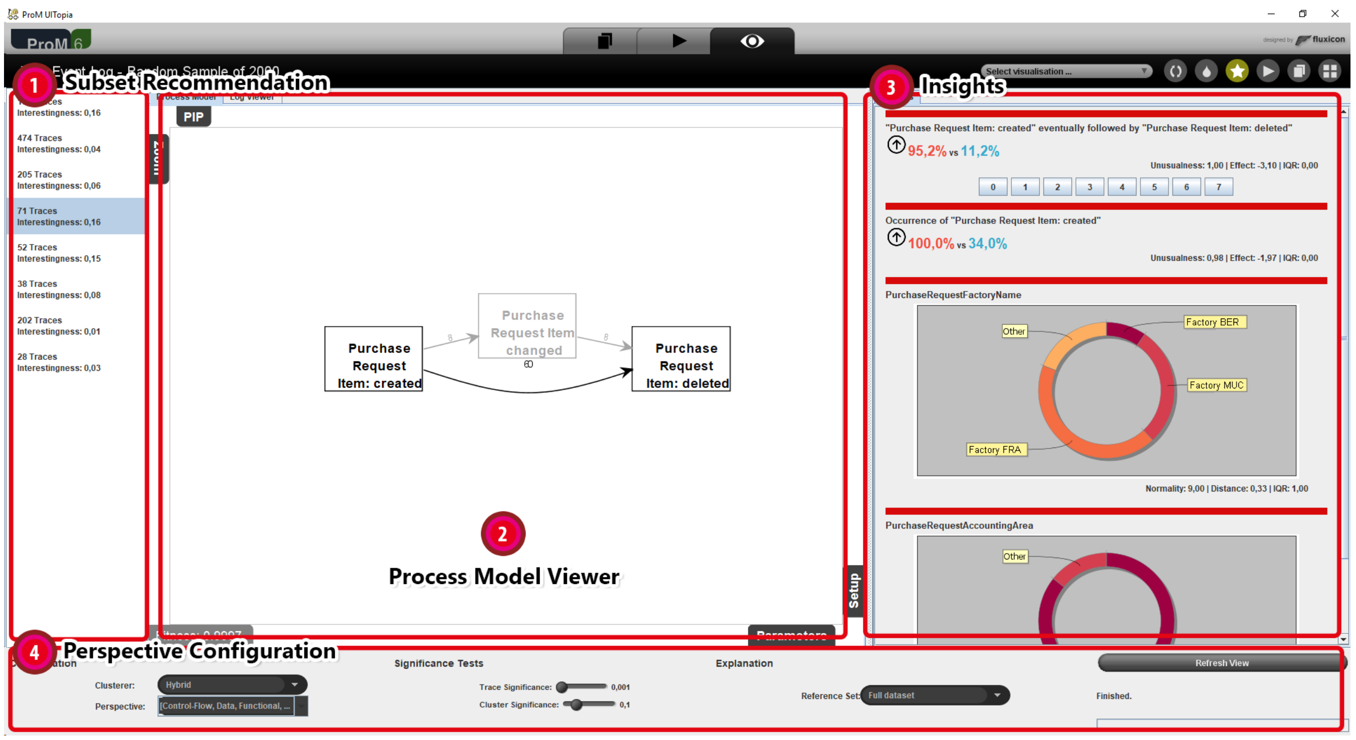
On the left side (1) the subset recommendations with the calculated interestingness score is shown from the trace clustering output. The recommendations are ranked by the interestingness and the number of process instances sharing similar behavior. After the analysts selects a subset recommendation, all the other components are updated immediately. The process explorer (2) shows a visual representation of the selected process instances using the HeuristicsMiner [24] for process discovery. Analysts can see how many instances have followed a specific path through the information system, reflected by the activities and transitions shown in the graph. Furthermore, all commonly used interactive browsing capabilities in process mining, like activity filtering, duration analysis and case viewer, are available. We decided to give the process model visualization a larger space because analysts tend to easier understand the actual process execution using a commonly used graph representation.

For further insights, the right side (3) shows the automatically extracted insights based on the selected process perspectives. We only show the significant insights as well as insights with at least an effect size of 0.2. Each insight is color-coded, showing the effect size of the deviation. This helps analysts to evaluate which of the measurements have a high deviation between the selected subset and the reference. The list also shows the measurement of the current selection and the reference event log, so the analyst can immediately see the difference. For better visualization, for some of the metrics an extended chart visualization is shown to the analyst (e.g., an histogram, a pie-chart or bar-chart).

Related insights are clustered together such that similar measurements are not shown multiple times. The user can still take a look at all the clustered insights by clicking on the buttons below a clustered insight.

5 EXPERIMENTS

We present a case study and a usage scenario to show how the recommendation engine and the produced insights help analysts during their exploratory analysis of event logs. For the case study we use a real life event log and analyzed it using our developed ProM plugin (see section 5.1). Furthermore, we showed our system a consulting company and discussed potential usage scenarios which we summarize in section 5.2.



Interestingness: 0,16
474 Traces
Interestingness: 0,04
205 Traces
Interestingness: 0,06
71 Traces
Interestingness: 0,16
52 Traces
Interestingness: 0,15
38 Traces
Interestingness: 0,08
202 Traces
Interestingness: 0,01
28 Traces
Interestingness: 0,03

"Purchase Request Item: created" eventually followed by "Purchase Request Item: deleted"
95,2% vs 11,2%
Unusualness: 1,00 | Effect: -3,10 | IQR: 0,00

Occurrence of "Purchase Request Item: created"
100,0% vs 34,0%
Unusualness: 0,98 | Effect: -1,97 | IQR: 0,00

PurchaseRequestFactoryName
Normality: 9,00 | Distance: 0,33 | IQR: 1,00

PurchaseRequestAccountingArea

Clusterer: Hybrid
Perspective: [Control-Flow, Data, Functional, ...
Trace Significance: 0,001
Cluster Significance: 0,1
Reference Set: Full dataset
Finished.

Figure 1: The ProcessExplorer plugin implemented into ProM. (1) The subset selection recommendations showing the number of traces and the interestingness measure. (2) A process model visualization of the selected subset. (3) The list of insights detected by ProcessExplorer, describing the deviation of means. (4) The configuration pane allows to select the process perspective to consider.

5.1 Procurement Process Case Study

For the case study we gathered a real life event log that was extracted from a SAP ERP system for procurement of goods. The resulting event log consists of a full month of data, 33.277 cases with 255.427 events and 37 different events. Each case consists – depending on the state of the case – of up to 15 different case attributes, describing for example, which good was ordered or which vendor delivered the purchased order. If the process discovery algorithm is applied to the whole event log, a spaghetti-like process model is discovered which is highly complicated and in-accurate.

We evaluated our system with the help of a process expert to see if the findings are actually valuable. During the analysis of the event log, the analyst also used a state-of-the-art process mining tool to validate the findings of ProcessExplorer. For the given event log, our system suggested 29 subset recommendations. In the following, we will present the most interesting findings.

First, we found that 6685 purchase orders were not made in the ERP system which leads to maverick buying (the process immediately starts with the invoice). Such behavior should be avoided because potential price discounts might be missed or required approvals were ignored. ProcessExplorer found this behavior and generated a separate cluster, ranked on the first position, only containing the cases that started with an invoice. Our system returned two insights for this cluster, describing that the number of involved resources is significantly lower and that the occurrence of the event *Vendor invoice: created* occurred in 100% of the cases whereas in average it occurred only 75.8% of the cases. Unfortunately, our system could not find any cluster insights which would explain the difference, for example, on the attribute values. However, it is still interesting to see that our system found that specific behavior.

The second cluster contained 3240 cases that were unfinished and ended with several different events. Interestingly, ProcessExplorer found the insight that number of cases containing the followed-by relation between *Purchase order: created* and *Goods receipt* was significantly lower with 35.4% than usual with 68.0%. Similar, the average total case duration was 2 days compared to 61 days normally. These insights help the expert to immediately see that these cases were not finished, and could be excluded for the further analysis.

Another interesting finding that our system as well as the analyst found was that 1115 cases finished with the deletion of the purchase request without ever creating an actual order. That exact finding was also extracted as a trace insight, discovering that the number of cases that contain the followed-by relation *Purchase request created* and *Purchase request deleted* was significantly higher than on average. Additionally, the number of events in these cases were lower, only containing 2.1 events compared to 7.0 in average. ProcessExplorer also found a cluster insight highlighting a significantly different distribution of procurement groups, for which this finding was specifically identified.

In summary, our system found interesting patterns that could also be confirmed by the expert using a state-of-the-art process mining tool. According to the analyst, the insights provided by the system immediately helped to understand the subset recommendation. Certain found patterns were not that interesting to further look at, but the analyst commented that clicking through the list of 29 subset recommendations is not very time consuming. Such false findings are not too much of an issue because the provided visualization quickly helped to identify them as uninteresting.

5.2 Usage Scenario: Exploring Unknown Event Logs

ProcessExplorer has many potential usage scenarios where the scope of a process analysis is not directly predefined. We also showed our system a consulting company which is specialized in process mining and the analysis of processes, and discussed possible usage scenarios. In the following, we present a usage scenario of ProcessExplorer for the exploratory analysis of business processes, based on our discussion.

In most process mining projects, the analyzed process is typically not known beforehand, thus the questions that are usually answered in such projects are very open. So one might want to know how the process looks in reality, where the bottleneck in the process is, or which department needs more resources to achieve a certain organization goal. One might also want to identify the suspicious cases or sequences in the process, find inefficiencies or deviations. Such open and generic questions can be answered using process mining in an exploratory fashion. The reconstructed process model is analyzed by an expert who uses his domain knowledge to identify interesting patterns. However, for event logs with large amount of data such open questions are hard to answer. ProcessExplorer can help in this scenario, where the exact analysis goal is not clearly defined and where the identification of interesting patterns is required. The analyst can use our recommender system to first identify the different process behaviors, ranked by interestingness. By clicking through the clusters, the analyst can get himself a good overview of the process and identify potential issues. Because clusters are ranked, the analyst will see the most interesting patterns first.

After the analyst has selected a certain process behavior, the discovered process model can be visually inspected. Besides the model, the analyst can also see the deviation of means for the selected measurements. Because the measurements are based on the selection of the different perspectives, the analyst will only see the ones that he is interested in. This gives him a clear view on the current observed process behavior. For example, if the duration between two activities are much higher than in average, the analyst may have a deeper look into the cases in a manual way. Besides the trace measurements, the analyst is also shown the cluster measurements which can be used to potentially find an explanation for the observed behavior. In our running example, the analyst may find that certain vendors are responsible for a long duration, thus the organization can take further action on resolving this issue. For instance, the analyst can use the selected subset of data and perform other analysis in the ProM framework to even find more valuable information.

We conclude that ProcessExplorer is an interactive exploratory system which makes browsing through large event logs easier, as it provides suggestions for further exploration. The calculated measurements help the analyst to evaluate the shown process model and to decide if further analysis is required.

6 RELATED WORK

Our work extends prior research on process mining, exploratory data interfaces, visualization tools, and visualization recommendations. In this section, we will give an overview of related work.

6.1 Trace Clustering and Subgroup Discovery

ProcessExplorer uses trace clustering to discover interesting subsets of data. A couple of different trace clustering methods [2, 4, 7, 9] were introduced. Most approaches use a classical data mining clustering method, like k-means or agglomerative hierarchical clustering, and define a similarity metric over the sequence of activities. Other methods use different metrics to optimize the outcome of the underlying discovered process models [23] or incorporate expert knowledge [11]. Delias et al. [5] describe a non-compensatory approach to overcome the issue of spaghetti-like models in flexible environments. The authors summarize the event log by combining multiple criteria into a single similarity metric. In [1] a process variant extraction is proposed which combines business rules based on decision points with process variants. A comparison framework for process instance clustering techniques has been presented in [17]. The authors also performed a systematic empirical analysis of the different clustering techniques. Subgroup discovery [15] can also be used to extract interesting patterns of a subset of process cases. The authors use existing class labels in the dataset to discover patterns for a particular target group and return common characteristics of that class. The quality of a subgroup is measured by interestingness measures.

6.2 Exploratory Search

The data input for process mining [18] is an event log which content is usually not known and from which valuable insights should be extracted. All process mining tools, like ProM [22], Celonis [21], Fluxicon Disco or QPR ProcessAnalyzer are exploratory tools, allowing a process expert to interactively browse and search through interesting subsets of the dataset. This is highly related to work on exploratory search [13] and data analysis (EDA). Typically, users do not know the characteristics of the specific dataset, nor the optimal way to achieve their goal.

The typical representation of an event log in process mining are process models (e.g., flow-charts, petri-nets, BPMN, or workflow-models), annotated with specific performance indicators (e.g., duration, throughput, etc.). Users can manually apply certain filters, highlight execution paths or compare the model with a reference (also known as conformance checking). A useful combination of business intelligence and process mining is implemented in Celonis [21], allowing to visually inspect data attributes and the process flow at the same time. Filtering on one visualization, for example, suppliers, automatically updates the flow-chart and vice-versa.

6.3 Visualization Recommendation

Voyager [26] and SEEDB [20] are two interactive browsing tools, recently developed for data tables and EDA. Both approaches support the user by providing visualization recommendations, reducing the amount of manual work to explore large datasets.

In process mining there exists only a limited number of approaches that deal with visualization recommendations. In [25] the user is pointed towards interesting parts of the process model by providing a linguistic summary, instead of showing a visual representation. Due to the high number of generated sentences, an optimized method is presented in [6]. A recommendation-based business process optimization (rBPO) was presented in [10]. rBPO generates action recommendations for currently running process instances by predicting metric deviations from former observations. Niedermann et al. [14] combine a catalog of formalized optimization techniques with operational data to assist analysts with the selection and application of optimizations.

To best to our knowledge there exists no process mining tool which points users towards interesting process behaviors, and recommends subsets of the data as well as visualizations. Furthermore, our approach extracts further valuable knowledge to characterize the recommendations and provide potential explanations.

7 CONCLUSION AND FUTURE WORK

In this paper, we introduced a novel recommendation system for subset selection and visual insights based on the four process perspectives. ProcessExplorer automatically analyzes the data and provides the user with interesting patterns, analytic insights and data explanations. These findings help to better understand large unknown event logs. We performed a case study, showing that our system found interesting insights for a real life event logs and provided a usage scenario, describing how ProcessExplorer can be adapted.

As a future work, we would like to further add more interactivity. Currently, the user is forced to select from a provided list of subset recommendation. But we imagine that the suggestions should also be dependent on the current manual data selection. Furthermore, the user should be able to modify or extend the selection. We are also aiming to add more cluster insights to better explain the current subset. Lastly, we are currently working on an extended evaluation of our system, comparing our method with the state-of-the-art experience in process mining tools.

Overall, we conclude that ProcessExplorer provides valuable support during the exploratory analysis of event logs in process mining.

ACKNOWLEDGMENTS

This project [01IS17050] is funded by the German Federal Ministry of Education and Research (BMBF) Software Campus project "AI-PM" .

REFERENCES

[1] Alfredo Bolt, Wil M. P. van der Aalst, and Massimiliano de Leoni. 2017. Finding Process Variants in Event Logs. In *On the Move to Meaningful Internet Systems. OTM 2017 Conferences*. Springer International Publishing, 45–52. https://doi.org/10.1007/978-3-319-69462-7_4

[2] R. P. Jagadeesh Chandra Bose and Wil M.P. van der Aalst. 2009. Context Aware Trace Clustering: Towards Improving Process Mining Results. In *Proceedings of the 2009 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 401–412. https://doi.org/10.1137/1.9781611972795.35

[3] J. Cohen. 2013. *Statistical Power Analysis for the Behavioral Sciences*. Taylor & Francis.

[4] Ana Karla Alves de Medeiros, Antonella Guzzo, Gianluigi Greco, Wil M. P. van der Aalst, A. J. M. M. Weijters, Boudewijn F. van Dongen, and Domenico Saccà. 2008. Process Mining Based on Clustering: A Quest for Precision. In *Business Process Management Workshops*. Springer Berlin Heidelberg, 17–29. https://doi.org/10.1007/978-3-540-78238-4_4

[5] Pavlos Delias, Michael Doumpos, Evangelos Grigoroudis, and Nikolaos Matsatsinis. 2017. A non-compensatory approach for trace clustering. *International Transactions in Operational Research* (2017). https://doi.org/10.1111/itor.12395

[6] Remco Dijkman and Anna Wilbik. 2017. Linguistic summarization of event logs – A practical approach. *Information Systems* 67 (2017), 114–125. https://doi.org/10.1016/j.is.2017.03.009

[7] Diogo Ferreira, Marielba Zacarias, Miguel Malheiros, and Pedro Ferreira. [n. d.]. Approaching Process Mining with Sequence Clustering: Experiments and Findings. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 360–374. https://doi.org/10.1007/978-3-540-75183-0_26

[8] G. Grahne and J. Zhu. 2005. Fast algorithms for frequent itemset mining using FP-trees. *IEEE Transactions on Knowledge and Data Engineering* 17, 10 (2005), 1347–1362. https://doi.org/10.1109/tkde.2005.166

[9] G. Greco, A. Guzzo, L. Pontieri, and D. Sacca. 2006. Discovering expressive process models by clustering log traces. *IEEE Transactions on Knowledge and Data Engineering* 18, 8 (2006), 1010–1027. https://doi.org/10.1109/tkde.2006.123

[10] Christoph Gröger, Holger Schwarz, and Bernhard Mitschang. 2014. Prescriptive Analytics for Recommendation-Based Business Process Optimization. In *Business Information Systems*. Springer International Publishing, 25–37. https://doi.org/10.1007/978-3-319-06695-0_3

[11] Pieter De Koninck, Jochen De Weerdt, and Seppe K. L. M. vanden Broucke. 2016. Explaining clusterings of process instances. *Data Mining and Knowledge Discovery* 31, 3 (2016), 774–808. https://doi.org/10.1007/s10618-016-0488-4

[12] David J. C. MacKay. 2003. *Information Theory, Inference and Learning Algorithms*. Vol. 53. Cambridge University Press. 628 pages. https://doi.org/10.1017/CBO9781107415324.004

[13] Gary Marchionini. 2006. Exploratory search. *Commun. ACM* 49, 4 (2006), 41. https://doi.org/10.1145/1121949.1121979

[14] Florian Niedermann and Holger Schwarz. 2011. Deep Business Optimization: Making Business Process Optimization Theory Work in Practice. In *Enterprise, Business-Process and Information Systems Modeling*. Springer Berlin Heidelberg, 88–102. https://doi.org/10.1007/978-3-642-21759-3_7

[15] Mohammadreza Fani Sani, Wil van der Aalst, Alfredo Bolt, and Javier García-Algarra. 2017. Subgroup Discovery in Process Mining. In *Business Information Systems*. Springer International Publishing, 237–252. https://doi.org/10.1007/978-3-319-59336-4_17

[16] Alexander Seeliger, Timo Nolle, and Max Mühlhäuser. 2018. Finding Structure in the Unstructured: Hybrid Feature Set Clustering for Process Discovery. In *Proceedings of the 16th International Conference on Business Process Management (BPM)*.

[17] Tom Thaler, Simon Ternis, Peter Fettke, and Peter Loos. 2015. A Comparative Analysis of Process Instance Cluster Techniques. *Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik (WI 2015)* August (2015), 423–437.

[18] Wil M. P. van der Aalst. 2011. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Vol. 136. Springer. 352 pages. https://doi.org/10.1007/978-3-642-19345-3

[19] Wil M. P. van der Aalst, Marcello La Rosa, and Flávia Maria Santoro. 2016. Business process management: Don't forget to improve the process! *Business & Information Systems Engineering* 58, 1 (2016), 1–6. https://doi.org/10.1007/s12599-015-0409-x arXiv:9780201398298

[20] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015. SeeDB. *Proceedings of the VLDB Endowment* 8, 13 (2015), 2182–2193. https://doi.org/10.14778/2831360.2831371

[21] Fabian Veit, Jerome Geyer-Klingeberg, Julian Madrzak, Manuel Haug, and Jan Thomson. 2017. The proactive insights engine: Process mining meets machine learning and artificial intelligence. *CEUR Workshop Proceedings* 1920 (2017).

[22] H. M. W. Verbeek, Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. 2011. XES, XESame, and ProM 6. In *Lecture Notes in Business Information Processing*. Springer Berlin Heidelberg, 60–75. https://doi.org/10.1007/978-3-642-17722-4_5

[23] Jochen De Weerdt, Seppe vanden Broucke, Jan Vanthienen, and Bart Baesens. 2013. Active Trace Clustering for Improved Process Discovery. *IEEE Transactions on Knowledge and Data Engineering* 25, 12 (2013), 2708–2720. https://doi.org/10.1109/tkde.2013.64

[24] a. J. M. M. Weijters, Wil M. P. van der Aalst, and a. K. Alves De Medeiros. 2006. Process Mining with the HeuristicsMiner Algorithm. *BETA Working Paper Series* 166 (2006), 1–34.

[25] Anna Wilbik and Remco M. Dijkman. 2015. Linguistic summaries of process data. In *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE. https://doi.org/10.1109/fuzz-ieee.2015.7337891

[26] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 649–658. https://doi.org/10.1109/tvcg.2015.2467191