Pariket: Mining Business Process Logs for Root Cause Analysis of Anomalous Incidents

Nisha Gupta[1], Kritika Anand[1], and Ashish Sureka[2]

[1] Indraprastha Institute of Information Technology, Delhi (IIITD), India
{nisha1345,kritika1339}@iiitd.ac.in
http://www.iiitd.ac.in/
[2] Software Analytics Research Lab (SARL), India
ashish@iiitd.ac.in
http://www.software-analytics.in/

Abstract. Process mining consists of extracting knowledge and actionable information from event-logs recorded by Process Aware Information Systems (PAIS). PAIS are vulnerable to system failures, malfunctions, fraudulent and undesirable executions resulting in anomalous trails and traces. The flexibility in PAIS resulting in large number of trace variants and the large volume of event-logs makes it challenging to identify anomalous executions and determining their root causes. We propose a framework and a multi-step process to identify root causes of anomalous traces in business process logs. We first transform the event-log into a sequential dataset and apply Window-based and Markovian techniques to identify anomalies. We then integrate the basic eventlog data consisting of the Case ID, time-stamp and activity with the contextual data and prepare a dataset consisting of two classes (anomalous and normal). We apply Machine Learning techniques such as decision tree classifiers to extract rules (explaining the root causes) describing anomalous transactions. We use advanced visualization techniques such as parallel plots to present the data in a format making it easy for a process analyst to identify the characteristics of anomalous executions. We conduct a triangulation study to gather multiple evidences to validate the effectiveness and accuracy of our approach.

Keywords: Anomalous Incidents, Business Process Mining, Decision Tree Classifier, Event Log, Markovian Based Technique, Root Cause Analysis.

1 Research Motivation and Aim

Business Process Management Systems (BPMS), Workflow Management Systems (WMS) and Process Aware Information Systems (PAIS) log events and activities during the execution of a process. Process Mining is a relatively young and emerging discipline consisting of analyzing the event logs from such systems for extracting knowledge such as the discovery of runtime process model (discovery), checking and verification of the design time process model with the

W. Chu et al. (Eds.): DNIS 2015, LNCS 8999, pp. 244–263, 2015.

runtime process model (conformance analysis) and improving the business process (recommendation and extension) [1]. A process consists of cases or incidents. A case consists of events. Each event in the event log relates to precisely one case. Events within a case are ordered and have attributes such as activity, timestamp, actor and several additional information such as the cost. The incidents and activities in event logs can be modeled as sequential and time-series data. Anomaly detection in business process logs is an area that has attracted several researcher's attention [2] [8]. Anomalies are patterns in data that do not conform to a well defined notion of normal behavior. Anomaly detection in business process logs has several applications such as fraud detection, identification of malicious activity and breakdown of the system and understanding the causes of process errors. Due to complex and numerous business processes in a large organizations, it is difficult for any employee to monitor the whole system. As a consequence of this anomalies occurring in a system remains undetected until serious losses are caused by it. Therefore, Root Cause Analysis (RCA) is done to identify root causes and sources of problems and improve or correct the given process so that major problems can be avoided in future.

The focus of the study presented in this paper is on anomaly detection in business process logs and identification of their root causes. We present a different and fresh perspective to the stated problem and our work is motivated by the need to extend the state-of-the-art in the field of techniques for anomaly detection and RCA in business process event logs. While there has been work done in the area of anomaly detection and RCA in business process logs, to the best of our knowledge, the work presented in this paper is the first focused study on such a dataset for the application of anomaly detection and RCA. The research aim of the work presented in this paper is the following:

1. To investigate Window based and Markovian based techniques for detecting anomalies in business process event logs.
2. To apply machine learning techniques such as decision tree classifier to extract rules describing cause of anomalous behavior.
3. To interactively explore different patterns of data using advanced visualization techniques such as parallel plot.
4. To investigate solutions assisting a process analyst to analyze decision tree and parallel plot results, thus identifying root cause of anomalous incidents.
5. To demonstrate the effectiveness of our proposed approach using triangulation study[1]. We conduct experiments on a recent, large and real-world incident management data of an enterprise.

2 Related Work and Research Contributions

We conduct a literature review of papers closely related to the work presented in this paper. Calderón-Ruiz et al. propose a novel technique to identify potential

[1] http://en.wikipedia.org/wiki/Triangulation_(social_science)

causes of failures in business process by extending available Process Mining techniques [7]. They test their technique using several synthetic event logs and are able to successfully find missing or unnecessary activities, and failed behavioural patterns that differ from successful patterns either in the control flow or in the time perspective [7]. Heravizadeh et al. propose a conceptual methodology of root-cause analysis in business processes, based on the definition of softgoals (nonfunctional requirements) for all process activities, as well as correlations between these softgoals and related quality metrics [10]. Suriadi et al. propose an approach to enrich and transform process-based logs for Root Cause Analysis based on classification algorithms [13]. They use decision trees to identify the causes of overtime faults [13]. Vasilyev et al. develop an approach to find the cause of delays based on the information recorded in an event log [14]. The approach is based on a logic representation of the event log and on the application of decision tree induction to separate process instances according to their duration [14].

Bezerra et al. present some approaches based on incremental mining [15] for anomaly detection, but these algorithms cannot deal with longer traces and/or logs with various classes of traces [2]. Then, in order to deal with such constraints, they begin to develop other solutions based on process mining algorithms available in ProM[2] framework [3] [6]. Bezerra et al. propose an anomaly detection model based on the discovery of an "appropriate process model" [6]. Bezerra et al. apply the process discovery and conformance algorithms from ProM framework for implementing the anomaly detection algorithms [3]. Bezerra et al. present three new algorithms (threshold, iterative, and sampling) to detect "hard to find" anomalies in a process log based only on the control-flow perspective of the traces [5]. This work does not deal with anomalous executions of processes that follow a correct execution path but deal with unusual data, or are executed by unusual roles or users, or have unusual timings [5]. Bezerra et al. develop an algorithm more efficient than the Sampling Algorithm [4]. They propose an approach for anomaly detection which is an extension of the Threshold Algorithm also reported in [3] [5], which uses process mining tools for process discovery and process analysis for supporting the detection [4].

In context to existing work, the study presented in this paper makes the following novel contributions:

1. Detection of anomalous traces in business process event-logs using Window-based and Markovian-based techniques (after transforming the event-log into a sequential) dataset.
2. Root Cause Analysis (RCA) of anomalous traces using parallel coordinate plots. Application of parallel coordinate plots for visualizing the characteristics of anomalous and normal traces (representing the traces and their attribute values as a polyline with vertices on the parallel axes).
3. Application of tree diagrams as a visual and analytical decision support tool for identifying the features of anomalous traces, thereby assisting a process analyst in problem solving and Root Cause Analysis (RCA).

[2] ProM is a pluggable and open-source framework for Process Mining.

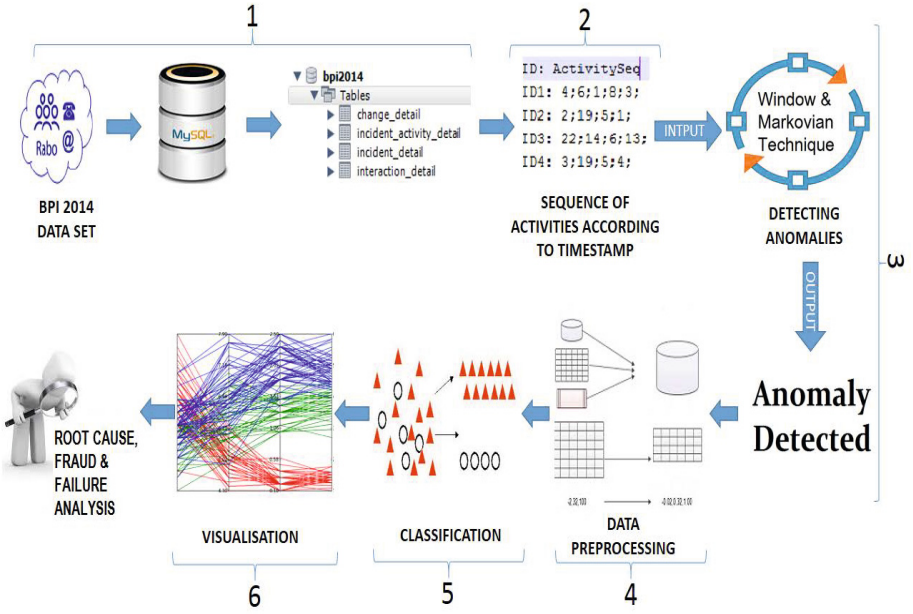

Fig. 1. Architecture diagram and data processing pipeline for Pariket (Mining Business Process Logs for Root Cause Analysis of Anomalous Incidents)

4. An in-depth and focused empirical analysis on a real-world dataset (Rabobank Group[3]: Activity log for incidents) demonstrating the effectiveness of the proposed approach. Application of triangulation technique to validate the outcome of RCA through cross-verification.

3 Research Framework and Solution Approach

Figure 1 shows the high-level architecture diagram of the proposed solution approach (called as *Pariket*). The proposed approach is a multi-step process primarily consists of 6 phases: experimental dataset collection, sequential dataset conversion, anomaly detection, data pre-processing, classification and visualization. The six phases are labeled in the architecture diagram in Figure 1. In phase 1, we download large real world data from Rabobank Group (refer to Section 4 on experimental dataset). The dataset consists of event logs from interactions records, incidents records, incident activities and change records. We choose incidents from incident activities to find out anomalous incident patterns. In phase 2, for a particular incident we order the type of activities according to increasing

[3] `http://data.3tu.nl/repository/uuid:c3e5d162-0cfd-4bb0-bd82-af5268819c35`

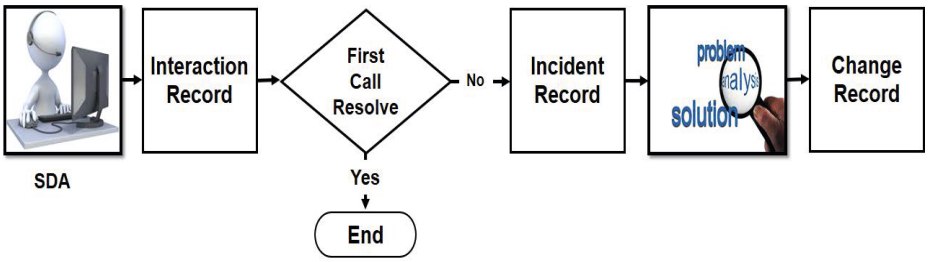

Fig. 2. Information Technology Infrastructure Library (ITIL) process implemented in Rabobank Group

order of DateTime Stamp. Each incident consisting of several activities is represented as a sequence of symbols (refer to Section 5.1 on experimental results). Each unique activity is mapped to a integer symbol. There are 39 different kinds of activities in the dataset and hence there are 39 different symbols. Some of the example of activities are: Referred (28), Problem Closure (22), OOResponse (18), Dial-In (10) and Contact Change (8). The sequences are of different length. The incidents with their corresponding sequence of activities serve as input to anomaly detection algorithms described in Section 5.2. In phase 3, we implement Window based and Markovian based technique [8] to detect anomalous incidents (refer to Section 5.2 on experimental results). We receive top N anomalous incidents as output from anomaly detection algorithms. We apply decision tree classifier and visualization techniques to identify root causes of anomalous incidents. Input to these techniques requires data to be in a particular format and of high quality. Hence, in phase 4 we perform data pre-processing to bring the data in the required format (refer to Section 5.3 of experimental results). In phase 5, we create decision tree using $J48$ algorithm in Waikato Environment for Knowledge Analysis (Weka)[4] (refer to Section 5.4). The $J48$ tree classifier is the $C4.5$ implementation available in Weka. $J48$ handles both numeric and nominal attribute values. In phase 6, we apply advanced visualization technique such as parallel plot in Tibco Spotfire[5] to interactively explore different regions of data (refer to Section 5.5). Business process analyst then analyze decision tree and parallel plot results to identify the root cause of anomalous incidents.

4 Experimental Dataset

We conduct our study on a large real world data from Rabobank Group Information and Communication Technology (ICT). The data is related to the Information Technology Infrastructure Library (ITIL) process implemented in the Bank. The ITIL process depicted in Figure 2 starts when an internal client

[4] http://www.cs.waikato.ac.nz/ml/weka/
[5] http://spotfire.tibco.com/

reports an issue regarding disruption of ICT service to Service Desk Agent (SDA). SDA records the complete information about the problem in Interaction Record. If the issue does not get resolved on first contact then a Incident record is created for the corresponding Interaction else the issue is closed. There can be many to one mapping between Interaction Record and Incident Record. If a issue appears frequently then a request for change is initiated.

The dataset is provided in the CSV format. It contains the event logs from interactions records, incidents records, incident activities and change records. The provided dataset is of six month duration from October 2013 - March 2014. Interactions that were not resolved before 31 March, were removed from the dataset. The attributes of original .CSV files are converted to appropriate data types, such as standardized timestamp formats, for analysis. After loading the data on to MySQL database, we build four tables: Interaction_detail, Incident_detail, Incident_activity_detail and Change_detail.

1. Interaction_detail - It has 147,004 records, each one corresponding to an interaction. Every record contains information like InteractionID, Priority, Category, Open Time, Close Time, Handle Time, and First Call Resolution (whether SDA was able to resolve the issue on first contact or not).
2. Incident_detail - It has 46,606 records, each one corresponding to an incident case. Every record has attributes like IncidentID, Related Interaction, Priority, Open Time, Handle Time, Configuration Item Affected etc.
3. Incident_activity_detail - It has 466,737 records. Each record contains an IncidentID with the activities performed on it. It also contains information about the Assignment Group that is responsible for a particular activity.
4. Change_detail - It contains records of the activities performed on each change case. It has information about Configuration Item Affected, Service Component Affected, Change Type and Risk Assessment etc.

As an academic, we believe and encourage academic code or software sharing in the interest of improving *openness and research reproducibility*. We release our code and dataset in public domain so that other researchers can validate our scientific claims and use our tool for comparison or benchmarking purposes (and also reusability and extension). Our code and dataset is hosted on GitHub[6] which is a popular web-based hosting service for software development projects. We select GPL license (restrictive license) so that our code can never be closed-sourced.

5 Experimental Results

We perform a series of steps to identify the root cause of anomalous incidents. Each of the following 6 sub-sections describes the steps consisting of procedure or approach and findings.

[6] `https://github.com/ashishsureka/pariket`

5.1 Sequential Dataset Conversion

We analyze all the tables and amongst them we choose Incident_activity_detail table to find out the anomalous incident patterns. This table contains a log of activities performed by the service team(s) to resolve incidents which are not resolved by first contact. The main reason for choosing this table is because it has information regarding the type of activities performed on a particular incident id and also the timestamp when this incident activity type started to resolve the issue.

The attribute IncidentActivity_Type represents the type of activity performed on the incident. There are 39 unique activities. Some of the examples are: Assignment (ASG), Status Change (STC), Update (UPD), Referred (REF), Problem Closure (PC), OOResponse (OOR), Dial-In (DI) and Contact Change (CC). Figure 3 represents the pareto chart showing the distribution of activities and their cumulative count. The Y-axis is in logarithmic scale. We assign integer number starting with 0 to 38 to these activities, and then we add an extra column IncidentActivity_Type_Number into the table Incident_activity_detail denoting this activity number. For a particular incident we order the activities according to increasing order of DateTime Stamp. This is done for all the unique incidents in the Incident_activity_detail table.

We apply Window Based and Markovian Based Techniques for detecting anomalous incidents. The input dataset to these algorithms has to be in sequential format. Therefore, to accomplish this we create a new table 'IncidentActivitySequence' containing two attributes 'IncidentID' and 'IncidentAcitvity_Type_Number'. Each record in this table contains all unique IncidentID's and sequences of IncidentActivity_Type_Number separated by semicolon according to timestamp from Incident_activity_detail. For example, corresponding to IncidentID 'IM0000012', the sequence of activities are '34;27;2;34;4;5;'.

5.2 Anomaly Detection

The outcome of the Section 5.1 is a list of all incident id's with their corresponding sequence of activities ordered according to timestamp. The aim of algorithms described in this Subsection is to identify anomalous incidents based on the obtained discrete sequences. There is no reference or training database available containing only normal sequences. Hence, our task is to detect anomalous sequences from an unlabeled database of sequences. The problem is of unsupervised anomaly detection. A formal representation of the problem is [8]: Given a set of n sequences, $S = \{S_1, S_2, ...,S_n\}$, find all sequences in S that are anomalous with respect to rest of S. This unsupervised problem can be solved by using a semi-supervised approach where we treat the entire dataset as training set and then score each sequence with respect to this training set. We assume that majority of sequences in the unlabeled database are normal as anomalies are generally infrequent in nature [8]. We use two algorithms, Window Based and Markovian Based described in the following Subsections for anomaly detection.

Algorithm 1: Window Based Algorithm (ID, S, k, λ, N)

Data: IncidentID (ID = $ID_1....ID_n$) and Sequence of activities (S = $S_1...S_n$) from table.
Result: Top N Anomalous IncidentID.

```
set windowSize = k, threshold = λ;
create a empty dictionary D, D'
create an arrayList anomalousIncidents;
foreach IncidentID ID_i in ID do
    S_i = get the sequence corresponding to ID_i
    set windowCount = S_i.length - windowSize + 1
    foreach j = 1 to windowCount do
        read the subsequence (W_j) of length = windowSize starting from j^th
        position in S_i
        if W_j is not present in D then
            add (W_j, 1) as (key, value) pair in D
        else
            add (W_j , value + 1) in D
foreach IncidentID ID_i in ID do
    S_i = get the sequence corresponding to ID_i
    set windowCount = S_i.length - windowSize + 1
    set anomalyScore = 0.0
    foreach j = 1 to windowCount do
        read the subsequence (W_j) of length = windowSize starting from j^th
        position in S_i
        get the (key, value) pair from D corresponding to key = W_j
        if value is less than threshold then
            anomalyScore = anomalyScore + 1
    anomalyScore = anomalyScore / windowCount
    add ID_i, anomalyScore into D'
sort D' according to decreasing anomalyScore
add top N IncidentId into anomalousIncidents
return anomalousIncidents
```

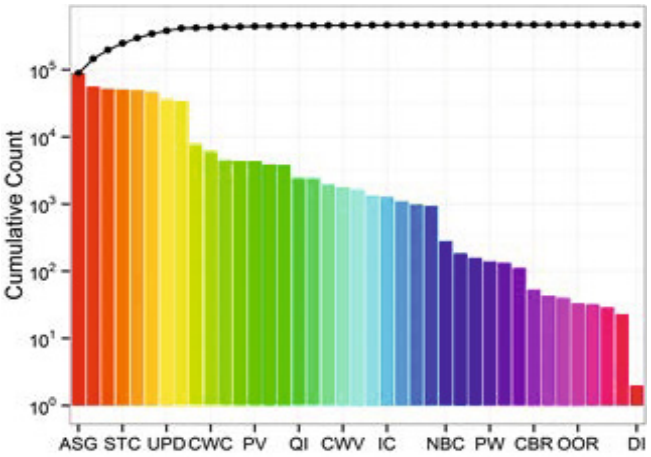

Fig. 3. Pareto chart showing the distribution of activities and their cumulative count. Y-axis is in logarithmic scale.

Window Based Technique. The motivation behind using window based technique is to determine anomalous sequences even if the cause of anomaly is localized to one or more shorter subsequences within the actual sequence [9]. Window based technique in general operates as, first we extract overlapping windows of fixed length (k) from a given test sequence. Then, we assign some anomaly score to each extracted window based on a threshold value (λ). Finally, the anomaly score of all the windows are combined to obtain an anomaly score for the test sequence [8].

The pseudocode for Window Based anomaly detection algorithm is shown in Algorithm 1. The input to the algorithm is data comprising of IncidentID, sequence of activities from table IncidentActivitySequence, window size (k), threshold (λ) and number of anomalous incidents (N). The algorithm returns top N anomalous IncidentID as output. The main challenge was to find out the size of window (k) and the value of threshold (λ). We analyze all the subsequences of window length less than 3. Our analysis reveals that they occur very frequently. Therefore, we cannot take them as anomalous subsequences because according to our previous assumption in Section 5.2 anomalies in our dataset are in minority. Therefore, k has to be equal to or greater than 3.

Algorithm 1 consists of two phases: training and testing. We choose 3 experimental parameters: $k = 3$, $\lambda = 4$ and $N = 1000$. The training phase is represented by Steps 4-12. During this phase, we obtain the sequence of activities for each IncidentID. From the sequence we extract k length overlapping (sliding) windows. We maintain each unique window with its frequency in normal dictionary D. The testing phase is represented by Steps 13-25. Every sequence of the training dataset is considered as the test sequence. During this phase, we extract sliding windows of length k from the test sequence S_i. A window W_j is assigned an

anomalyScore of 1 if the frequency associated with the window W_j in dictionary D is less than the threshold value (λ) else anomalyScore is 0. To calculate the anomalyScore of a complete test sequence S_i, we take summation of anomalyScore of all the subsequence windows contained in it. The anomaly score of the test sequence is proportional to the number of anomalous windows in the test sequence [11]. The result obtained after executing Steps 14-21 is then divided by the number of windows contained in the test sequence. This normalization is done to take into account the varying lengths of sequences. The anomalyScore of 1 for a test sequence denotes most anomalous and 0 as least anomalous. We store the IncidentID and its corresponding anomalyScore in the dictionary $D^{'}$. Then, we sort the IncidentID's in decreasing order of anomalyScore and return top N anomalous IncidentID's.

Markovian Based Technique. We apply fixed Markovian technique [8] which is based on the property of short memory of sequences. This property states that the conditional probability of occurrence of a symbol s_i is dependent on the occurrence of previous k symbols with in a sequence S_i [12]. The conditional probability of occurrence of a symbol s_i in a sequence S_i is given by Equation 1:

$$P(s_i|s_{(i-k)...s_{(i-1)}}) = \frac{freq(s_{(i-k)}...s_i)}{freq(s_{(i-k)}...s_{(i-1)})} \tag{1}$$

where $freq(s_{(i-k)} ...s_i)$ is the frequency of occurrence of the subsequence $s_{(i-k)}$ $...s_i$ in the sequences in S and freq$(s_{(i-k)}...s_{(i-1)})$ is the frequency of occurrence of the subsequence $s_{(i-k)}...s_{(i-1)}$ in the sequences in S.

The pseudocode for Markovian based anomaly detection algorithm is shown in Algorithm 2. The input to the algorithm is data comprising of IncidentID, sequence of activities from table IncidentActivitySequence, window size (k) and number of anomalous incidents (N). The algorithm returns top N anomalous IncidentID as output. Algorithm 2 consists of two phases: training and testing. Steps 4-17 represents the training phase. During this phase, we create two dictionaries D_k and D_{k+1} of length k and k+1 respectively. The process for creation of dictionary is similar to that described in Section 5.2. We choose $k = 3$ for our experiment. It takes into account the subsequences of length 4 which are dependent on previous 3 symbols. Steps 18-32 represents the testing phase. Steps 18-32 are repeated for each IncidentID in table IncidentActivitySequence. We extract the test sequence S_i corresponding to a IncidentID ID_i in Step 19 and calculate the number of subsequences of length k in Step 20. Steps 23-28 are repeated for each each subsequence within the test sequence S_i. Step 23 and 24 reads the subsequence W_j and W_{j+1} of length k and K+1 respectively starting from position j. Step 25 and 26 calculates the frequency $value_j$ and $value_{j+1}$ of W_j and W_{j+1} from the dictionaries D_k and D_{k+1}. We calculate the conditional probabilities of symbols in Step 27 by using Equation 1. We calculate the overall probability of S_i using the Equation 2:

Algorithm 2: Markovian Based Algorithm (ID, S, k, N)

Data: IncidentID (ID= $ID_1....ID_n$) and Sequence of activities (S= $S_1...S_n$) from table.
Result: Top N Anomalous IncidentID.

```
create a empty dictionary D_k, D_{k+1}, D'
create an arrayList anomalousIncidents
foreach IncidentID ID_i in ID do
    S_i = get the sequence corresponding to ID_i
    set noOfSubsequences = S_i.length - k + 1
    foreach j = 1 to noOfSubsequences do
        read the subsequence (W_j) of length = k starting from j^th position in S_i
        read the subsequence (W_{j+1}) of length = k+1 starting from j^th position in S_i
        if W_j is not present in D_k then
            add (W_j, 1) as (key, value) pair in D_k
        else
            add (W_j, value + 1) in D_k
        if W_{j+1} is not present in D_{k+1} then
            add (W_{j+1}, 1) as (key, value) pair in D_{k+1}
        else
            add (W_{j+1}, value + 1) in D_{k+1}
foreach IncidentID ID_i in ID do
    S_i = get the sequence corresponding to ID_i
    set noOfSubsequences = S_i.length - k + 1
    set anomalyScore = 0.0, prob = 0
    foreach j = 1 to noOfSubsequences - 1 do
        read the subsequence (W_j) of length = k starting from j^th position in S_i
        read the subsequence (W_{j+1}) of length= k+1 starting from j^th position in S_i
        get the (key_j, value_j) pair from D_k corresponding to key = W_j
        get the (key_{j+1}, value_{j+1}) pair from D_{k+1} corresponding to key = W_{j+1}
        r = (value_j) / (value_{j+1});
        prob = prob + log (r) ;
    prob = prob / noOfSubsequences
    TestSequenceProbability = e^prob
    anomalyScore = 1 / TestSequenceProbabity;
    add ID_i, anomalyScore into D'
sort D' according to decreasing anomalyScore
add top N IncidentID into anomalousIncidents
return anomalousIncidents
```

$$P(S_i) = \prod_{i=1}^{l} P(s_i|s_1 s_2 ... s_{i-1}) \quad (2)$$

where l is the length of the sequence S_i and s_i is the symbol occurring at position i in S_i [8]. For simplification, we take *log* on both the sides in Equation 2, the modified equation is used in the Step 28. We normalize the probability in Step 29 to take into account the varying length of sequences. We calculate anomaly score for test sequence S_i as the inverse of the probability of S_i in Step 31. Less probability of the test sequence means more anomaly score. We store the IncidentID and its corresponding anomalyScore in the dictionary $D^{'}$. Then, we sort the IncidentID's in decreasing order of anomalyScore and return top N IncidentID's.

Table 1. Name, Type and Description of Some of Attributes in Merged table of Interaction_detail and Incident_detail

Attribute Name	**Attribute Type**	**Description**
Incident_CIType(Aff)	Nominal	There are 13 distinct types of CIs. Example: software, storage, database, hardware, application.
Incident_CISubType(Aff)	Nominal	There are 64 CI Sub-types. Example : web based, client based, server based, SAP.
Incident_Priority	Nominal	There are 5 categories of priority i.e {1, 2, 3, 4, 5}.
Incident_Category	Nominal	There are 4 Incident Category i.e {Incident, Request For Information, Complaint, Request For Change}
Incident_OpenTime	Date Format	The Open Time of Incident is in 'yyyy-MM-dd HH:mm:SS' format.We convert into timestamp in 'hours.
Incident_HandleTime	Date Format	The Handle Time of Incident is in 'yyyy-MM-dd HH:mm:SS format.We convert into timestampin 'hours.
Interaction_CIType(Cby)	Nominal	There are 13 distinct types of caused by CIs. Example: software, storage, database, hardware, application.
Interaction_CISubType(Aff)	Nominal	There are 64 CI Sub-types. Example: web based, client based, server based, SAP.
Closure Code	Nominal	There 15 distinct types of Closure Code. Example: Unknown, Operator Error, Enquiry, Hardware, Software.
Anomalous	Nominal	{Yes,No}.

5.3 Data Pre-processing

We receive top N anomalous IncidentID's as output from algorithms described in Section 5.2. Our aim is to identify root causes of anomalous incidents. We apply data mining techniques (machine learning) such as decision tree classifier to extract rules describing anomalous behaviour. Input to these techniques requires data to be in a particular format and of high quality. Hence, we apply data pre-processing techniques to bring the data in the required format and also to improve the quality. The data pre-processing helps in improving the accuracy and efficiency of the subsequent mining processes. Data goes through series of steps during pre-processing phase: integration, cleaning and transformation etc.

First, we join two tables 'Interaction_detail' and 'Incident_detail' (refer to Phase 4 of architecture diagram in Figure 1). We use attribute 'RelatedIncident' from Interaction_detail table as foreign key and 'IncidentID' from Incident_detail as primary key to perform the join. We give new name 'Interaction_Incident' to the merged table. The merged table contains all the information for the issues that could not be resolved on first call. We create two copies of table Interaction_Incident: Interaction_Incident_Markovian and Interaction_Incident_Windows. We add new attribute 'Anomalous' to the newly created tables. We make the value of attribute 'Anomalous' as 'Yes' for all the top N anomalous IncidentID's and 'No' for rest of the records. Table 1 represents name, type and description of some of attributes obtained after merging Interaction_detail and Incident_detail. The anomalous IncidentID's for table Interaction_Incident_Windows are obtained from the outcome of Algorithm 1. The anomalous IncidentID's for table Interaction_Incident_Markovian are obtained from the outcome of Algorithm 2. We use J48 algorithm for classification using decision tree in Weka. The J48 algorithm handles missing values itself by replacing them with the most frequent observed non-missing values.

Next we transform open time, close time for Interaction and open time, reopen time, resolved time, closed time for Incident given in datetime format. We covert the datetime format that is 'yyyy-MM-dd HH:mm:SS' into timestamp in 'hours' to be useful for classification. For this, we take reference datetime as '1970-01-01 17:13:01'. Handle time for both Interaction and Incident is given in seconds in comma separated format. We transform it by removing comma because the J48 algorithm takes input in CSV or Attribute-Relation File Format (ARFF) format. The pre-processed data serves as input to the classification and visualization techniques.

5.4 Classification

Data mining technique such as decision tree offer a semi automated approach to identify root causes of anomalous incidents. Choosing a data mining analysis tool to execute decision tree algorithm can be a challenge. Popular open source data mining packages include Weka, R, Tanagra, Yet Another Learning Environment (YALE), and Konstanz Information Miner (KNIME)[7]. We choose Weka as it

[7] `http://www.sciencedirect.com/science/article/pii/S0272271207001114`

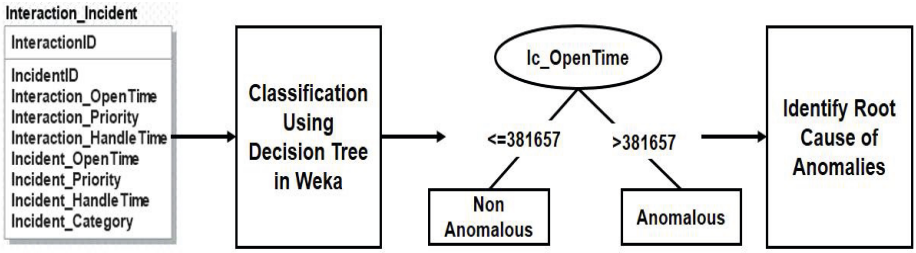

Fig. 4. Flow of classification using Weka

provides a flexible interface which is easy to use. Weka is open source software issued under the GNU General Public License. Weka is a collection of machine learning algorithms for data mining tasks like data pre-processing, classification, clustering, association rules, visualization, etc.

Figure 4 depicts overall flow of classification using decision tree in Weka. The pre-processed data which we obtain after data pre-processing in Section 5.3 serves as input to Weka. The input to Weka is normally in CSV or ARFF format. We use Attribute Selector to select attributes in Weka. Attribute selection involves searching through all possible combination of attributes in the data to find which subset attributes works best for prediction. We perform classification using decision tree algorithm in Weka. Decision tree offers many benefits: easy to understand by user, handles variety of input data such as nominal and numeric and handles missing values in dataset. We apply $J48$ algorithm for decision tree based classification on data. The $J48$ tree classifier is the $C4.5$ implementation available in Weka. The $J48$ builds decision tree from a set of labeled training data using the concept of information entropy. We change the default parameters in $J48$ algorithm like binarySplits, ConfidenceFactor, minNumObj, etc. But, there is no improvement observed in the results. Result is displayed in classifier output window. To view tree in graphical format click on 'visualize tree' option in pop menu.

We consider combination of attributes or parameters as root causes of anomalous incidents which occur on the path from the root to leaf showing anomalous as Yes. Figure 5 and 6 shows the fragments of the decision tree extracted from Weka (due to limited space it is not possible to display the entire tree). To represent figures more clearly, Incident is written as Ic and Interaction is written as Ir. We create decision tree in Figure 5a and 5b by applying $J48$ algorithm on records from Interaction_Incident_Markovian in CSV format. We observe that there are 240 anomalous incidents whose incident open time is greater than 381657 (hrs). Figure 5b shows that there are 38 anomalous incidents whose interaction open time is greater than 383499 (hrs). Decision tree in Figure 6 is for results from Interaction_Incident_Windows in CSV format. Figure 6 depicts there are 67 anomalous incidents whose interaction open time is greater than 384871 (hrs) and incident open time is greater than 384822 (hrs). We obtain

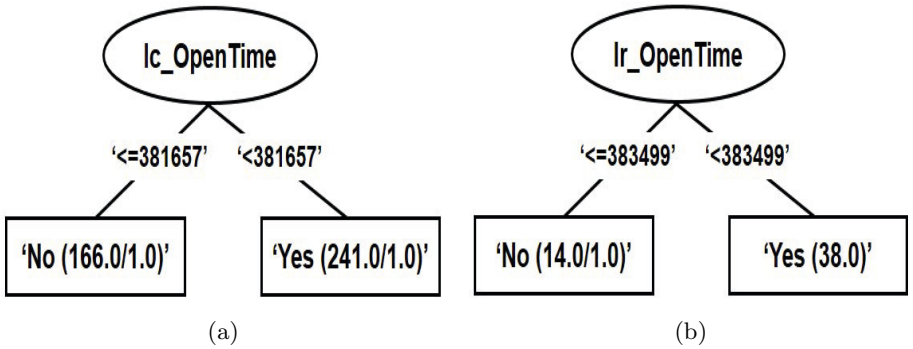

Fig. 5. Fragments of decision tree using Weka based on anomalous incidents received from Markovian based technique

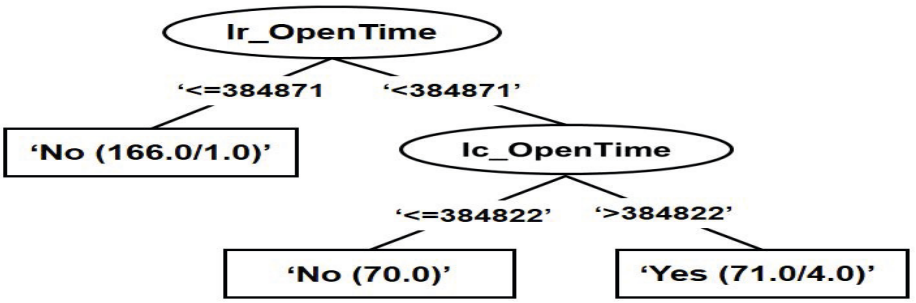

Fig. 6. Fragment of decision tree using Weka based on anomalous incidents received from Window based technique

attributes Incident_OpenTime, Interaction_OpenTime, Incident_Priority and Incident_Category on the path which leads to anomalous incident leaf nodes. And, there is a path consisting of only Incident_OpenTime which classifies 240 incidents as anomalous. Therefore, open time of incidents alone or combination of open time of interaction, open time of incident, priority of incident and category of incident are causes behind anomalous incidents.

5.5 Visualization

Visualization techniques are used to facilitate user interaction with data. User analyze data by carefully examining it and using different tools on it. Visualization techniques help to identify usual trends and anomalies which are present in data. To achieve this, We use the Tibco's Spotfire platform. Tibco Spotfire is an analytics software that helps quickly uncover sights for better decision making. It is used to detect patterns and correlations present in the data that were hidden in our previous approach using Decision tree. Among many features provided by Spotfire, we use Parallel Coordinate Plot for visualization.

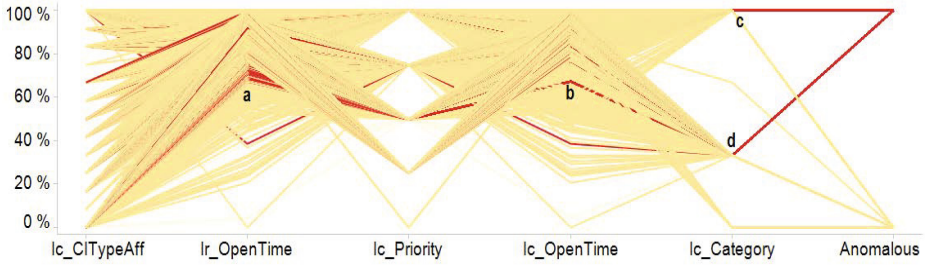

Fig. 7. Parallel coordinate plot depicting the behaviour of incident CI type affected, interaction open time, incident priority, incident open time and incident category for anomalous and non-anomalous incidents from Markovian technique

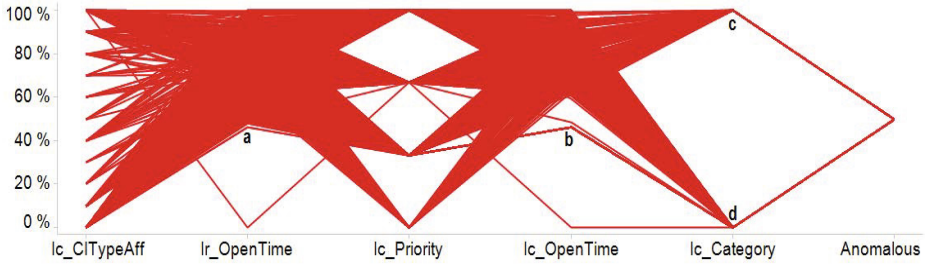

Fig. 8. Parallel coordinate plot depicting the behaviour of incident CI type affected, interaction open time, incident priority, incident open time and incident category for anomalous incidents from Markovian technique

Parallel Coordinate Plot maps each row in the data table as a line. Each attribute of a row is represented by a point on the line. The values in Parallel Plot are normalized. It means lowest value for an attribute in the column is 0% of entire data values in that column while the highest value is 100% unlike the line graphs. Therefore, we cannot compare the values in one column with the values in other column. The data fed into parallel plot is the integrated data which we obtain after the pre-processing phase described in Section 5.3. We create parallel plot by following four steps.

1. Load the data into the Tibco Spotfire. Data can be of type Spotfire Binary Data Format (SBDF), TXT, XLSX, CSV. etc.
2. Choose Parallel Coordinate Plot from Insert tab.
3. Select attributes from column option of the properties section. The selected attributes will be displayed on X-axis of plot.
4. Color the lines or profile of each data row depending on attribute value.

We create plots by taking into account different combination of attributes along with the attribute Anomalous (Yes/No). Figure 7 represents patterns of the attributes: Ic_CITypeAff, Ir_OpenTime, Ic_Priority, Ic_OpenTime and

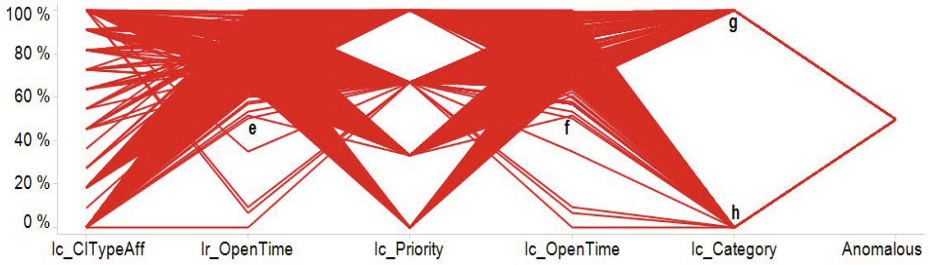

Fig. 9. Parallel coordinate plot depicting the behaviour of incident CI type affected, interaction open time, incident priority, incident open time and incident category for anomalous incidents from Window based technique

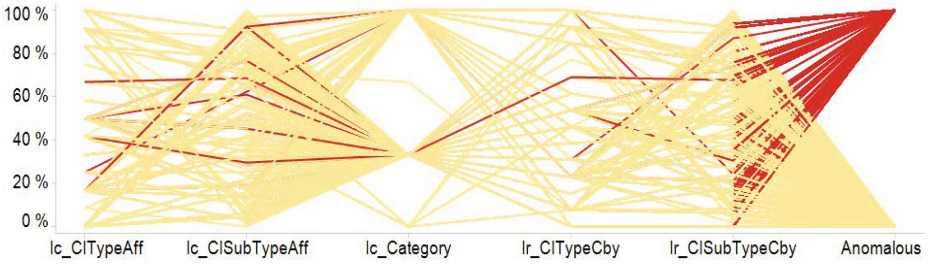

Fig. 10. Parallel coordinate plot depicting the behaviour of incident CI type affected, incident CI subtype affected, incident category, interaction CI type affected and interaction CI subtype affected for anomalous and non-anomalous incidents from Markovian technique

Ic_Category for the complete dataset on parallel plot. The dataset consists of records from table Interaction_Incident_Markovian created in Pre-processing phase in CSV format. Due to scarcity of space, Incident is written as Ic and Interaction is written as Ir. We show anomalous incidents with red color and non-anomalous with yellow color. We consider only anomalous incidents in Figure 8 for better clarity. The attributes Ic_CITypeAff and Ic_Priority individually do not show any useful information regarding anomalous behavior of incidents. Anomalous Incidents are falling in all the Ic_CITypeAffs and Ic_Prioritys, therefore they alone cannot be cause of anomalies. Majority of anomalous incidents are lying above ‘a for the attribute Ir_OpenTime. According to it for all anomalous incidents, Ir_OpenTime is above 376305 (hrs) and majority of them have Ir_OpenTime above 381658 (hrs). Point ‘b shows that majority of anomalous incidents Ic_OpenTime above 381658 (hrs). Points ‘c and ‘d depicts that out of 4 Ic_Category : Incident, Complaint, Request for Information and Request for Change, anomalous incidents fall into only 2 categories: Incident and Request for Information. Figure 9 shows parallel plot for the dataset obtained from table Interaction_Incident_Windows. We consider only records which have

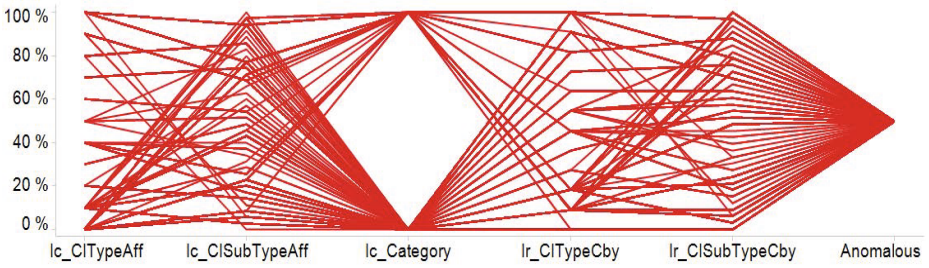

Fig. 11. Parallel coordinate plot depicting the behaviour of incident CI type affected, incident CI subtype affected, incident category, interaction CI type affected and interaction CI subtype affected for anomalous incidents from Markovian technique

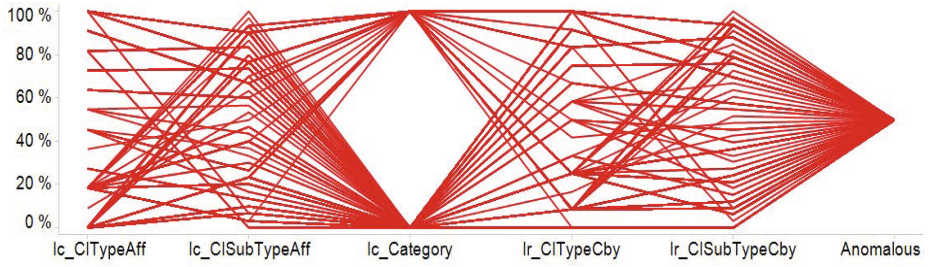

Fig. 12. Parallel coordinate plot depicting the behaviour of incident CI type affected, incident CI subtype affected, incident category, interaction CI type affected and interaction CI subtype affected for anomalous incidents from Window based technique

value of attribute Anomalous as 'YES'. Figure 9 shows that majority of anomalous incidents are lying above 'e for the attribute Ir_OpenTime. According to it for all anomalous incidents, Ir_OpenTime is above 37082 (hrs) and majority of them have it above 381512 (hrs). Point 'f shows that majority of anomalous incidents Ic_OpenTime above 381512 (hrs). Points 'g and 'h depicts that out of 4 Ic_Category's anomalous incidents fall into only 2 categories: Incident and Request for Information.

We choose attributes Ic_CITypeAff, Ic_CISubTypeAff, Ic_Category, Ir_CITypeAff and Ir_CISubTypeAff for Figure 10, 11 and 12. Figure 10 represents parallel plot for complete dataset with anomalous incidents obtained from Markovian technique. Figure 11 and 12 represents parallel plot only for anomalous incidents obtained from Markovian and Window based technique respectively. The selected attributes do not show any useful information regarding anomalous behavior of incidents. Anomalous Incidents are falling in all values for Ic_CITypeAff, Ic_CISubTypeAff, Ir_CITypeAff and Ir_CISubTypeAff. Therefore, they cannot be the cause of anomalies. It concludes that Affected Configuration Item's (CI)

type and subtype do not influence cause of anomaly. The Affected Configuration Item is the CI where a disruption of ICT service is noticed. The attribute Ic_Category is showing the same behavior as in Figure 7, 8 and 9.

By comparing the parallel plots for Markovian and Window based technique, it is evident that anomalous incidents from both the techniques follow the same patterns.

5.6 Triangulation Study

In this Subsection, we present our approach on validating the uncovered root cause. In our experiments, we use a publicly available dataset and we do not have facts to validate the root-cause. The real source of the problem is confidential and not known to us or publicly available. We apply data triangulation technique consisting of gathering evidences from multiple sources to validate the root cause[8]. Data triangulation is a well-known technique and we believe is well-suited for our study. We define two evaluators: outcome from parallel coordinate plots and output of decision trees on the dataset. Our objective is to investigate if the findings and indicators from the two different evaluators converge to the same conclusion. Decision tree results described in Section 5.4 show that root causes of anomalous incidents are open time of incidents alone or combination of open time of interaction, open time of incident, priority of incident and category of incident. Visualization using parallel plot depicts that cause of anomalies is not dependent on Affected Configuration Item's (CI) type and subtype which confirms with decision tree results. The parallel plot results also show that root cause of anomalies is dependent on open time of incident, open time of interaction and category of incident (refer to Section 5.5). The experimental results from the decision tree are in agreement with the parallel plot results, thereby validating our approach.

6 Conclusion

We present a novel approach for identification of anomalous traces and executions from event-logs generated by Process Aware Information Systems (PAIS) and a new technique for Root Cause Analysis (RCA) of anomalous traces. The key components of the proposed framework are: anomaly detection from sequential dataset using Window-based and Markovian-based technique, extraction of rules and characteristics of anomalous traces using decision-tree classifiers and application of parallel co-ordinate plots to visualize distinctions between anomalous and normal traces. We conduct a series of experiments on real-world dataset and conduct a triangulation study to demonstrate that the proposed approach is effective. Experimental results reveal agreement in output from Window-based and Markovian technique increasing the confidence in the classification result. We observe that data pre-processing and transformation is needed and impacts the outcome of parallel coordinate plot and decision tree classifier.

[8] http://en.wikipedia.org/wiki/Triangulation_(social_science)

References

[1] Van der Aalst, W.: Process mining: discovery, conformance and enhancement of business processes (2011)

[2] Bezerra, F., Wainer, J.: Anomaly detection algorithms in logs of process aware systems. In: Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 951–952. ACM (2008)

[3] Bezerra, F., Wainer, J.: Fraud detection in process aware systems. International Journal of Business Process Integration and Management 5(2), 121–129 (2011)

[4] Bezerra, F., Wainer, J.: A dynamic threshold algorithm for anomaly detection in logs of process aware systems. Journal of Information and Data Management 3(3), 316 (2012)

[5] Bezerra, F., Wainer, J.: Algorithms for anomaly detection of traces in logs of process aware information systems. Information Systems 38(1), 33–44 (2013)

[6] Bezerra, F., Wainer, J., van der Aalst, W.M.P.: Anomaly detection using process mining. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. LNBIP, vol. 29, pp. 149–161. Springer, Heidelberg (2009)

[7] Calderón-Ruiz, G., Sepúlveda, M.: Automatic discovery of failures in business processes using process mining techniques

[8] Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection for discrete sequences: A survey. IEEE Transactions on Knowledge and Data Engineering 24(5), 823–839 (2012)

[9] Forrest, S., Hofmeyr, S., Somayaji, A., Longstaff, T.: A sense of self for unix processes. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, pp. 120–128 (May 1996)

[10] Heravizadeh, M., Mendling, J., Rosemann, M.: Root cause analysis in business processes (2008)

[11] Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. Journal of computer security 6(3), 151–180 (1998)

[12] Ron, D., Singer, Y., Tishby, N.: The power of amnesia: Learning probabilistic automata with variable memory length. Machine Learning 25(2-3), 117–149 (1996)

[13] Suriadi, S., Ouyang, C., van der Aalst, W.M., ter Hofstede, A.H.: Root cause analysis with enriched process logs. In: Business Process Management Workshops, pp. 174–186 (2013)

[14] Vasilyev, E., Ferreira, D.R., Iijima, J.: Using inductive reasoning to find the cause of process delays. In: 2013 IEEE 15th Conference on Business Informatics (CBI), pp. 242–249. IEEE (2013)

[15] Wainer, J., Kim, K.-H., Ellis, C.A.: A workflow mining method through model rewriting. In: Fukś, H., Lukosch, S., Salgado, A.C. (eds.) CRIWG 2005. LNCS, vol. 3706, pp. 184–191. Springer, Heidelberg (2005)