Discovering Signature Patterns from Event Logs

R.P. Jagadeesh Chandra Bose
Eindhoven University of Technology
The Netherlands 5600 MB
Email: j.c.b.rantham.prabhakara@tue.nl

Wil M.P. van der Aalst
Eindhoven University of Technology
The Netherlands 5600 MB
Email: w.m.p.v.d.aalst@tue.nl

***Abstract*—More and more information about processes is recorded in the form of so-called "event logs". High-tech systems such as X-ray machines and high-end copiers provide their manufacturers and services organizations with detailed event data. Larger organizations record relevant business events for process improvement, auditing, and fraud detection. Traces in such event logs can be classified as desirable or undesirable (e.g., faulty or fraudulent behavior). In this paper, we present a comprehensive framework for discovering *signatures* that can be used to explain or predict the class of seen or unseen traces. These signatures are characteristic patterns that can be used to discriminate between desirable and undesirable behavior. As shown, these patterns can, for example, be used to predict remotely whether a particular component in an X-ray machine is broken or not. Moreover, the signatures also help to improve systems and organizational processes.**

Our framework for signature discovery is fully implemented in ProM and supports class labeling, feature extraction and selection, pattern discovery, pattern evaluation and cross-validation, reporting, and visualization. A real-life case study is used to demonstrate the applicability and scalability of the approach.

Key words: Process Mining, Signature Patterns, Event Log, Discriminatory Patterns

I. Introduction

Many of today's information systems record an abundance of event logs. Such event logs often contain data indicating the health of a process or the status of a case, etc. One can consider such health indicators as class labels. For example, an X-ray machine event log might contain information on system failures and broken parts/components; an insurance claim event log might contain information on whether a claim is fraudulent or not. Organizations are interested in gaining further insights on such health indicators such as learning whether there are any common patterns among the cases with a certain class label or whether there are any discriminatory patterns between cases of different classes. *Signature discovery* is concerned with finding such patterns.

Signature discovery is a *process mining* technique [1]. Process mining aims to discover, monitor and improve real-life processes by extracting knowledge from event logs readily available in today's (information) systems. Signature discovery examines traces in such event logs and aims to diagnose differences and predict the class of unclassified traces. There are many applications for signature discovery. We mention two motivating examples:

- *Fault diagnosis of high-tech systems:* High-tech systems such as medical devices, copier machines, and wafer scanners, all generate event logs capturing their day-to-day operations [2]. These systems may malfunction when they are used abnormally (operational processes deviating significantly from their normal/intended usage). Malfunctions are also noticed when parts/components in the system encounter faults and/or deteriorate. System event logs are often the primary source of information for diagnosing (and predicting) the causes of failures in these systems. Early detection and diagnosis of system malfunctions can help avoid catastrophic failures and reduce productivity loss. For large and complex systems such as these, there is a pressing need for better techniques for processing, understanding, and analyzing these data [3].
- *Detecting fraudulent claims:* Insurance companies across all sectors (e.g., healthcare, automobile, property, etc.) are plagued by fraudulent claims costing billions of dollars annually [4]. Detecting fraud and abuse relies heavily on analysts/auditors inspection of claims in conjunction with domain knowledge. Automated fraud detection is only viable if complex patterns can be uncovered in massive amounts of low-level data [5]. There is a need for analytical techniques for effective detection of fraud [6], [7], [8], [9]. Assuming that there exists a historical database where we have "cases" comprising the evidence collected so far that indicates fraud, one can try to learn patterns/characteristics of behavior in such cases that discriminate them from normal behavior and use the uncovered patterns for monitoring future instances.

In this paper we present a framework for automated discovery of signature patterns from event logs where some or all cases carry a label indicating the class that they belong to. We evaluate the goodness of this framework on a real-life case study on finding patterns that can be correlated to part replacements in an X-ray machine.

The rest of the paper is organized as follows. Section II discusses our framework for signature discovery while Section III presents the realization of the framework. A real-life case study of discovering signature patterns for diagnosing faults in X-ray machines is presented in Section IV. Related work is presented in Section V. Finally, Section VI concludes the paper.

II. Signature Discovery Framework

We propose the framework depicted in Fig. 1 for discovering patterns that discriminate between different classes of

978-1-4673-5895-8/13/$31.00

behavior. Starting point is an *event log* consisting of events. Each event refers to an activity or action (i.e., a well-defined step in some process) and is related to a particular *case* (i.e., a *process instance*). The events belonging to a case are *ordered*. Therefore, cases are represented as traces of events that correspond to "runs" of a possibly unknown process. Events may have all kinds of attributes (e.g., timestamp, resources used, temperature, costs, etc.). The proposed framework in Fig. 1 is generic and works for any *event log* with labeled cases (signifying different classes of behavior) with a provision for some of the cases remaining unlabeled. We now explain the constituents of our signature discovery framework.

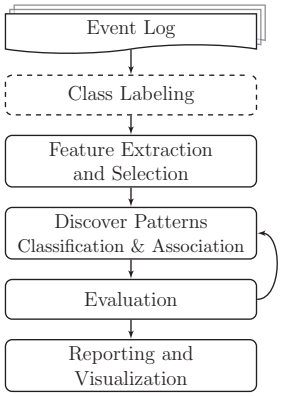

Fig. 1. Framework for signature discovery. The block depicted in dashed rectangle is an optional step that is to be considered when some of the cases in an event log are unlabeled.

A. Class Labeling

When event logs contain some cases that are unlabeled, an important question to address is *How can we assign labels to those unlabeled instances?* Efficient means to automatically or semi-automatically derive labels need to be designed. We propose the use of clustering and/or classification techniques, such as the k-nearest neighbor [10] and one-class support vector machines (SVM) [11], in machine learning to assist in class labeling. For example:

- If the unlabeled instances are to be assigned one of the class labels already present in the event log, then one may consider the k-nearest neighbor approach. The basic idea is to determine the k-nearest labeled instances for each of the unlabeled instances and assign the majority class of the k instances as the class label for the unlabeled instance.
- If the labeled instances in the event log belong to only one class and we are interested in labeling the unlabeled instances to utmost two-classes, e.g., fraudulent and non-fraudulent as in the case of insurance claims, an interesting approach is the use of one-class support vector machines. Here, we assume that the instances of one-class (e.g., non-fraudulent) are labeled. One-class SVMs work with the assumption that all positive (non-fraudulent) instances are alike while each negative (fraudulent) instance can be negative in its own way, i.e., the distribution of the negative instances is unknown. Once a one-class SVM is built over the non-fraudulent instances, any unlabeled instance can be evaluated to either belonging to the non-fraudulent class or not and labeled accordingly.

After the execution of this step, *all instances in the event log should have a class label*. After this preprocessing step, we can discover patterns that are specific for each class and discriminatory between the classes.

B. Feature Extraction and Selection

This step corresponds to extracting the features from an event log, which form the basis for signature patterns. Once features are defined, each instance in the event log is to be transformed into a vector space where the elements of the vector correspond to the value of the selected feature in the instance. We argue that a wide variety of feature types need to be considered and the choice of the feature type largely depends on the nature of the problem and its manifestation in the event log. Domain knowledge can assist us in selecting an appropriate feature. We recommend the consideration of *individual events*, *sequence features* (tandem arrays, maximal repeats and its variants), and *alphabet features* defined in [12], [13] as features. Sequence features are important when an occurrence of a particular sequence of events in the system log defines a symptomatic pattern, e.g., when a part malfunctions, the components that depend on/interact with this faulty part *retries* and seeks for a response from the part. Retries often manifest as *loops*, which are captured with tandem arrays [12], [13]. As discussed in [13], alphabet features are derived from sequence features by relaxing the ordering of events. Sequence features that are defined over the same set of activities (events) are considered to be equivalent under an alphabet feature. In addition to the above features, one may also consider features catering to other perspectives such as data (e.g., data objects and their values in each trace).

If the number of features extracted is large, then it leads to the problem of *curse of dimensionality* [14]. Feature selection techniques deal with removing irrelevant and redundant features. One can adopt simple filtering techniques such as removing infrequent features to advanced dimensionality reduction techniques such as principal component analysis [15] for feature selection. Once the feature extraction and selection is done, we transform the event log into a vector space as depicted in TABLE I.

C. Discover Patterns

Given a dataset as depicted in TABLE I, the goal of this step is to discover the patterns over the features, which are strongly correlated to the class label (e.g., normal or faulty). We adopt standard data mining techniques, i.e., decision tree learning [16], [17] and association rule mining [18], [19]. These two learning algorithms are chosen primarily for three reasons:

- they are non-parametric, i.e,. no specific data distribution (of the input dataset) is assumed

TABLE I

THE LABELED CASES IN AN EVENT LOG ARE TRANSFORMED INTO A VECTOR SPACE BASED ON THE CHOSEN FEATURES $(f_1, f_2, \ldots, f_m)$. ONE CAN CHOOSE BETWEEN A NOMINAL (BINARY) REPRESENTATION (WHERE THE VALUE FOR A FEATURE IN A CASE CORRESPONDS TO THE PRESENCE/ABSENCE OF THE FEATURE IN THAT CASE) AND A NUMERIC REPRESENTATION (WHERE THE VALUES CORRESPOND TO THE FREQUENCY OF THE FEATURE IN THE CASE).

Instance	f_1	f_2	...	f_m	Class
1	3	1	...	0	**N**
2	0	6	...	1	**F**
3	1	0	...	4	**F**
⋮	⋮	⋮	⋮	⋮	
n	2	2	...	0	**N**

- they generate simple, understandable rules that are easy to interpret by domain experts
- they can easily handle imbalanced datasets, i.e., datasets where the instances of each class are not approximately equally represented

For the association rule mining, we adopt the special subset called the *class association rules* [19], which is an integration of classification rule mining and association rule mining. We do not present the details of these algorithms in this paper. The interested reader is referred to [16], [17], [18], [19], [14].

The result of this step are rules such as:

If $f_1 \geq v_{11}$ AND $f_3 = v_{31}$ AND $f_7 = v_{72}$ then **F** OR
If $f_2 = v_{26}$ AND $f_4 = v_{47}$ then **N** OR
If $f_5 = v_{50}$ then **F**

where the v_{ij}'s are the values for the corresponding features.

D. Evaluation

We adopt standard metrics in data mining such as the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), and derived metrics from these such as accuracy, sensitivity, specificity, precision, and F1-score to evaluate the goodness of the discovered signatures. Models with sensitivity and specificity close to 1.0 are preferred [14].

For a given dataset, one can build many classifiers. The differences mainly stem from the choice of parameter values for the learning algorithm (e.g., split criterion in decision trees, minimum support and minimum confidence constraints in association rule mining, etc.). An important characteristic of any learned model is its *generalizability*. Generalization refers to the performance of a learned model over unseen examples [14]. If the entire dataset is used for learning the signatures, then the uncovered signatures may be *overfitting*. As a result, the learned model may perform well on the input dataset, but performs poorly on unseen examples. Therefore, we adopt cross-validation techniques during the learning phase in the above step.

Cross-validation [14], [20] is a model selection technique where the input dataset is divided into two subsets, viz., a *training set* and a *validation set*. The model is learned on the training set and evaluated on the validation set. A special case of cross-validation is the k-folds cross validation technique where the input dataset is split into k subsets, and the model is learned on the training data comprising of $k-1$ subsets and validated on the last subset. This is repeated k times with k different splits between the training and validation data. The cross-validation performance is the average of the results (with respect to metrics such as accuracy) on all the splits. We prefer signature patterns with a better cross-validation performance. If the performance is not satisfactory, one may change the parameter settings for the learning algorithm and re-learn the signatures.

E. Reporting and Visualization

The last step in the framework reports the findings and visualizes the results. Automated reports eliciting the signature patterns along with their performance metrics are generated. Apart from reports, one may depict the results in pictorial forms such as pie-charts and scatter plots. For example, Fig. 2 depicts the projection of a two-class multi-dimensional data onto the top two principal components obtained using principal component analysis [15]. Such a visualization helps in assessing the goodness of a feature set. In the figure, we can see that the two classes (normal and faulty) are clearly separable thereby indicating that the chosen feature set representation for the cases is good enough to find discriminatory patterns.

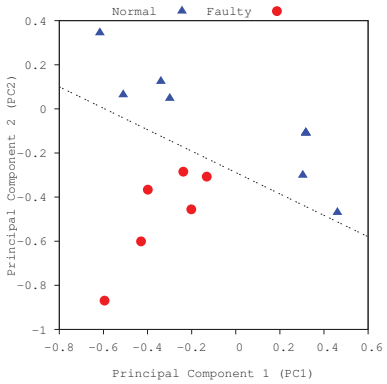

Fig. 2. Visualization of dataset using principal components.

III. IMPLEMENTATION

We have implemented the framework presented in the previous section as the 'Signature Discovery' plug-in in ProM[1]. Given an event log where the cases are labeled (indicating different classes of behavior), this plug-in uncovers discriminatory patterns that distinguish between the different classes of behavior. This plug-in assumes that the label of a case is provided as an attribute value with the key "Class" in the event log. Fig. 3 depicts the configuration step for class labeling while Figs. 4 and 5 depict the configuration steps for feature extraction/selection and learning algorithm respectively. Fig. 6 shows the results provided by the plug-in.

[1]ProM is an extensible framework that provides a comprehensive set of tools/plug-ins for the discovery and analysis of process models from event logs. See http://www.processmining.org for more information and to download ProM.

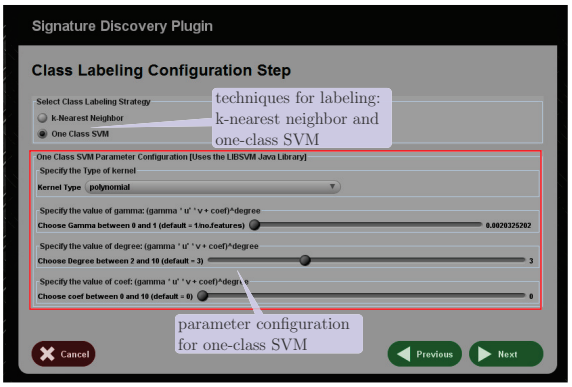

Fig. 3. Configuration step for labeling unlabeled instances in an event log. The plug-in supports two algorithms, viz., k-nearest neighbor and one-class SVM for class labeling.

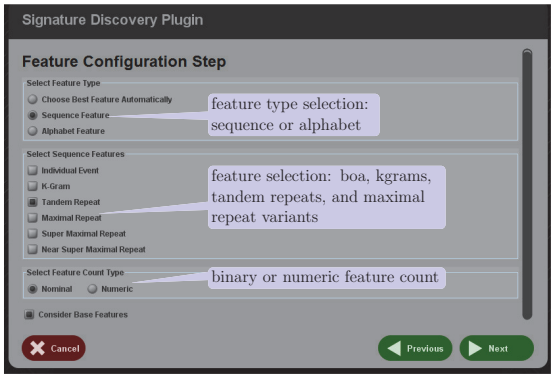

Fig. 4. Configuration step for feature extraction and selection. Different types of features are supported, e.g., sequence and alphabet features: tandem arrays, maximal repeats and variants, and individual events.

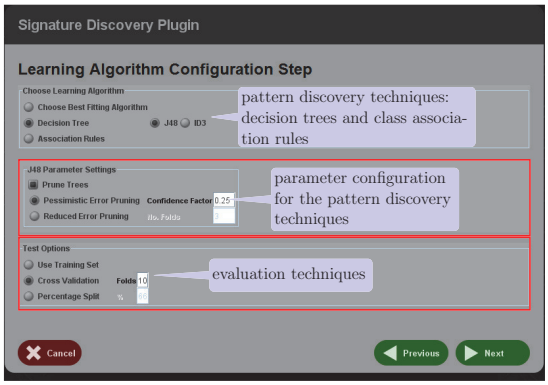

Fig. 5. Configuration step for the learning algorithm for discovering signature patterns. Two classes of algorithms, viz., decision trees and association rule mining are supported.

IV. Case Study

In this section, we present the case study of fault diagnosis of X-ray machines from Philips Healthcare, a global leader in professional and consumer healthcare. Although it is undesirable for these systems to malfunction, in reality, these systems

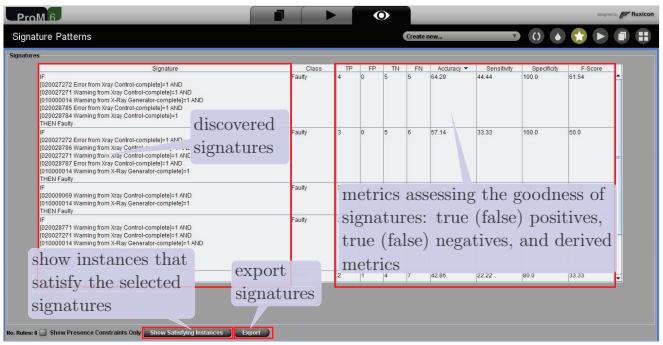

Fig. 6. Results of the 'Signature Discovery' plug-in. The plug-in estimates different quality metrics for each of the discovered signatures.

do malfunction during their lifetime. However, when they do, it is important that these problems are quickly and predictably corrected. The X-ray machines considered in this study are installed across the globe and continuously log all major events (e.g., system operations, warnings, errors, etc.). Moreover, problems (customer complaints) and the actions performed to rectify them are logged as job sheets. The combination of both data sources (logs and job sheets) provide a rich source of historical service data. The organization sees an opportunity of improving their system maintenance through *log-based fault diagnosis*. More specifically, *they are interested in investigating whether the diagnostic value of system logs can be improved by discovering patterns that can be correlated to known problems and/or corrective actions with high confidence*. In this case study, *we confine ourselves to the task of finding symptomatic patterns in the event logs that can be associated to a malfunction requiring the replacement of parts in an X-ray machine*. Parts that can be replaced in the system are called *Field Replaceable Units* (FRUs).

A. Data Selection

The data selection process starts with first choosing the FRU we are interested in, e.g., FRUs for which the variation in mean-time-to-repair (MTTR) is large. This FRU could have been replaced in many systems as part of corrective maintenance in the past. We can identify all such systems from the job sheets database. Furthermore, each system can have multiple calls associated with this FRU replacement, i.e., it could be the case that the same part had to be replaced several times on a particular system at different periods of time. Since the system could have undergone version upgrades, it is recommended (by domain experts) that the (system, call) pairs are segregated based on their versions. Each call is associated with a call open date and a call close date. Furthermore, the system event logs are recorded every day. For each call, we consider logs from the corresponding system a few days before the call open date and a few days after the close date for analysis. The rationale is that if there exists a symptomatic pattern, it should have manifested in the system logs prior to and during the life time of the complaint and that they disappear in the event logs after the part replacement. The

number of days that one should consider before (after) the call open (close) date is largely dependent on the nature of the FRU and is to be chosen on a trial and error basis guided by domain knowledge. For example, a malfunction in a critical part such as an X-ray tube is noticed immediately whereas a malfunction in hard disk may not be noticed immediately. Thus it may be sufficient to consider just a couple of days before/after the call open/close date for the X-ray tube while for the hard disk, a larger time window is recommended.

B. Defining Cases–Scoping and Filtering

During a single day of machine operation, the system could have been (re)started or shutdown multiple times. A session of system's operation constitutes the sequence of events during the normal operation mode between startup and shutdown as illustrated in Fig. 7. Sessions form the basis for defining a case. The events that are recorded in the X-ray system are

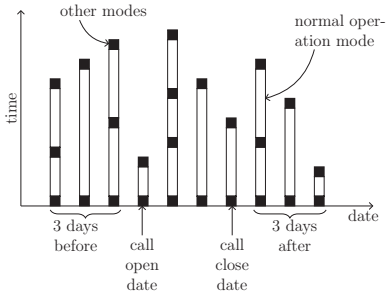

Fig. 7. Events during normal operation mode signify the events during the regular usage of the system and constitute the focus of analysis. The system could have been restarted multiple times during a single day. Each sequence of events during the normal operation mode surrounded by other system modes defines a session. In this example, we consider the log files 3 days before/after the call open/close date of a part replacement in a particular system.

very fine-grained. This makes the total number of events that are logged in a single day/session quite large, in the order of a few thousands. Identifying the symptomatic patterns pertaining to the malfunction of a FRU in the fine-grained event logs log is a challenging task. This can be attributed to the fact that the events that potentially bear an indication of the abnormality form a small fraction of the overall event data. Considering the whole log can induce a huge amount of unrelated events thereby making the task of signature discovery analogous to searching for a *needle in a haystack*. Domain experts suggest that a malfunction in a FRU reflects as error and/or warning events in the log pertaining to the component (unit) it belongs to and/or components with which it interacts with. Accordingly, we pre-process the log as follows:

- for a given FRU for which we are interested in identifying the symptomatic patterns in the log, we first identify (based on domain knowledge) the units (components) that are related to the FRU, e.g., if X-ray tube is chosen as the FRU, the units related to this are X-ray Control, X-ray Generator, and Geometry.
- only the error/warning events pertaining to the units related to the FRU during a session are considered

As mentioned earlier, the symptomatic patterns are expected to have manifested in the system logs prior to and during the life time of the complaint, i.e., on/before the call close date, and disappear in the event logs after the part replacement (call close date). Therefore, our problem of signature discovery is to uncover patterns consistent across the different calls (pertaining to that part replacement), which appear only in the event logs prior to the corresponding call close dates and disappear in the event logs after the call close date. It is important to note that the manifestation of patterns pertaining to a problem occurs only when that functionality or behavior is invoked on the system. In other cases, we see a normal behavior of the system. Hence in the time-period prior to the call-close date (i.e., the time period between which the customer sees some abnormality and the time at which the problem is supposed to have been resolved), it is quite possible that the system reflects a normal behavior during some of the sessions. However, it is unknown which sessions exhibit normal behavior.

We propose a means of transforming the system logs to labeled cases. For this case study, we expect the cases to be labeled as normal (N) and faulty (F). We use the *juxtaposed sessions* approach to transform system logs into labeled cases. The *juxtaposed sessions* approach creates two cases per call. The sessions on/before the call close date are all appended into a single case with a distinct delimiter (i.e., special characters/symbols out of the activity alphabet) between them and labeled as faulty (F). Similarly, the sessions after the call close date are appended together with a distinct delimiter between them and labeled as Normal (N). The distinct delimiter is essential to ensure that patterns do not overlap across sessions during the discovery process. Fig. 8 depicts this approach.

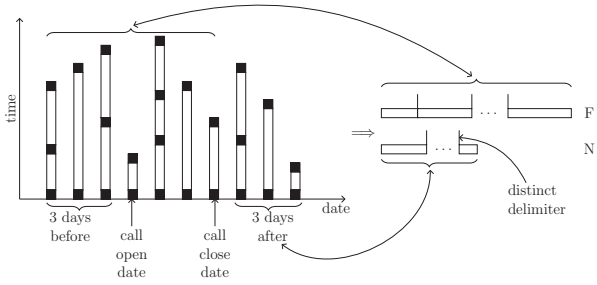

Fig. 8. Scenario where each call defines two cases. The sessions on/before the call close date are juxtaposed and assigned the label 'faulty' (F) whereas the case defined by the juxtaposed sessions after the call close date can be considered to be 'normal' (N). Delimiters are used to distinguish the boundaries between sessions.

At the end of this process, we have an event log with cases that carry a class label indicating a particular class of behavior. We now proceed to uncovering the signature patterns in the next section.

C. Signature Discovery

In this section, we discuss the application of the proposed framework in uncovering signatures for two of the FRUs,

which we anonymize as FRU I and FRU II. First, we discuss the results for FRU I. For learning the signature patterns, we considered this FRU's replacements that happened in the years 2008 and 2009. As discussed earlier, since different versions of the system can have different signatures, we split the systems according to their versions and discover the signatures for each version separately[2]. From the jobsheets, we identified the systems and the dates when this FRU was replaced and selected system log files between three days before the call open date and three days after the call close date for these systems. We considered the error/warning events from three units, which we anonymize (for confidentiality reasons) as Unit A, Unit B, and Unit C. These three units are considered to be the most relevant for this FRU by the domain experts. We used the juxtaposed sessions approach for defining the cases and class labels. Using this procedure, we created cases with two labels, i.e., normal or faulty, for each system type and version, e.g., there are 32 instances for the system version 4.3.5.

We discovered the signature patterns from these instances using the framework described in Section II. We used a combination of tandem repeat alphabet and maximal repeat alphabet features [13] in conjunction with the class association rules for learning the signature patterns. A couple of anonymized signatures for the faulty class are provided in TABLE II. The two signatures differ in the last two events. The interpretation for this is that this FRU has multiple failure modes and the manifestation of failure modes differs in the system event logs. Each of the signatures in TABLE II captures one of these failure modes.

TABLE II
ANONYMIZED SIGNATURE PATTERNS FOR FRU I.

If	xxxxxxxx1 Warning from Unit A is Present AND xxxxxxxx2 Error from Unit A is Present AND xxxxxxxx4 Warning from Unit B is Present AND xxxxxxxx4 Warning from Unit A is Present AND xxxxxxxx5 Error from Unit A is Present	Then	Faulty
If	xxxxxxxx1 Warning from Unit A is Present AND xxxxxxxx2 Error from Unit A is Present AND xxxxxxxx4 Warning from Unit B is Present AND xxxxxxxx6 Warning from Unit A is Present AND xxxxxxxx7 Error from Unit A is Present	Then	Faulty

We have evaluated the goodness of the signature patterns on an independent test set of system logs between Jan 2010 and Jun 2011 (Note that the signatures were discovered using logs from 2008 and 2009). Signatures of a particular system type and version are evaluated against systems of the same type and version. TABLE III depicts the performance of the signatures for two different versions of systems. The interpretation of the true positives, false positives, true negatives, and false negatives are as follows:

- *TP:* the signature is present in one or more log instances[3] of a system AND there is a FRU replacement in the system subsequently AND the signature disappears after the replacement,
- *FP:* the signature is present in one or more log instances of a system BUT there is no FRU replacement in the system (OR) the signature is present in one or more log instances of a system AND there is a FRU replacement in the system subsequently BUT the signature does not disappear after the replacement,
- *TN:* the signature is not present in a log instance of a system AND there is no FRU replacement in this system, and
- *FN:* the signature is not present in a log instance of a system BUT there is a FRU replacement in the system.

False negatives indicate that the discovered signatures are not complete and that there might be other symptomatic patterns which we are not able to uncover. This results when the training data does not represent all manifestations of failure modes of the FRU. False negatives are not devastating whereas false positives are. False positives have serious repercussions and need to be minimized. False positives can lead to false replacements. False negatives affect the sensitivity and F1-score metrics while false positives affect the specificity, precision, and F1-score metrics.

From TABLE III, we can see that the uncovered signatures perform quite well with a very high accuracy (above 98%). Note that our evaluation involved a large set of independent systems (743 in number) with logs considered in a different time period from that of the training data. The uncovered signatures for version 3.1.7 are able to detect all but one of the required replacements and there are no false positives. The discovered signatures for version 4.3.5 are able to indicate a problem in this FRU for 24 of the 32 replacements and could not capture the rest 8 replacements. The part could have exhibited a failure mode different from that of the captured signatures in these 8 replacements. Furthermore, we see just two false positives in this case.

As another example, we considered a different FRU, anonymized as FRU II. Just like in the previous scenario, we considered part replacements in 2008 and 2009. Event logs from systems with this part replacement were used for learning the signature patterns. Signatures discovered using the class association rule mining algorithm on the individual events (as the feature set) performed better when compared to other features and learning algorithms on the training data. We evaluated the uncovered signatures on an independent set of system logs between Jan 2010 and Jun 2011. TABLE III depicts the performance of the discovered signatures for two different versions of systems. We can see that, even in this case, the uncovered signatures perform good as reflected by the high accuracy (above 98%). The uncovered signatures for version 3.1.7 are able to detect all but one of the required replacements and there are no false positives. As mentioned before, false negatives potentially indicate different failure modes, the signatures of which are not captured in our discovery phase, primarily due to lack of representative instances

[2]It could be the case that for two different versions, the signatures are the same (this implies that there is no change between these versions with respect to this FRU)

[3]a log instance is one session of the system log in the normal operation mode

TABLE III
PERFORMANCE OF THE DISCOVERED SIGNATURES FOR THE TWO FRU REPLACEMENTS ON AN INDEPENDENT TEST SET OF SYSTEMS. VARIOUS METRICS SUCH AS THE TRUE POSITIVES (TP), FALSE POSITIVES (FP), TRUE NEGATIVES (TN), FALSE NEGATIVES (FN), AND DERIVED METRICS SUCH AS ACCURACY, SENSITIVITY, SPECIFICITY, PRECISION, AND F1-SCORE ARE MEASURED.

Part	Version	No. Systems	TP	FP	TN	FN	Accuracy $\frac{TP+TN}{\text{No.Systems}}\%$	Sensitivity (r) $\frac{TP}{TP+FN}\%$	Specificity $\frac{TN}{TN+FP}\%$	Precision (p) $\frac{TP}{TP+FP}\%$	F1-score $\frac{2*p*r}{p+r}\%$
FRU I	3.1.7	120	3	0	116	1	99.16	75.00	100.00	100.00	85.71
	4.3.5	623	24	2	589	8	98.39	75.00	99.66	92.30	82.75
FRU II	3.1.7	120	7	0	112	1	99.16	87.50	100.00	100.00	93.33
	4.3.5	623	47	2	564	10	98.07	82.45	99.64	95.92	88.67

for this failure mode in the training phase. Systems of version $4.3.5$ had a larger number of FRU II replacements and the discovered signatures are able to detect a problem in the FRU II in 82% of the cases (sensitivity metric). The discovered signatures resulted in only two false positives, but do not cover all failure modes. This is reflected by the 10 false negatives. As mentioned earlier, false negatives are less problematic than false positives. The signatures for these failure modes also can be discovered when event logs representing these failure modes are provided in the training dataset. To summarize, the proposed framework for signature discovery shows significant promise in fault diagnosis of X-ray machines.

The discovered signatures can be added to a knowledge base and the logs of systems scanned for the presence of signature patterns. This can assist the Field Service Engineers (FSEs) during their diagnostic efforts. The FSEs can be provided with log analyzer tools that check for the presence of signatures. The manifestations of signature patterns suggest potential problematic parts (FRUs) corresponding to the signature. Since diagnostics are considered to be the most time consuming and the most difficult task, such an automated assistance is expected to reduce the MTTR significantly.

V. RELATED WORK

Any framework or methodology that attempts at discovering signature patterns, which discriminate between classes of behavior, is bound to use machine learning/data mining techniques. The differences between the solutions mainly stem from the nature of application/domain, input data and its treatment, and the definition and scope of patterns. The problem of signature discovery can essentially be viewed as one aimed at inducing a classifier for an event log with labeled traces. Folino et al. [21] have proposed a decision tree based predictive model defined over a set of attributes. The approach that we present in this paper is generic and is based on standard classical machine learning techniques: decision trees [16], [17] and association rule mining [18], [19]. However, our approach differs from [21] in four ways: (i) we start from event logs rather than tabular data, (ii) in addition to decision trees, our approach also considers association rules between attributes and the class labels, (iii) our approach also addresses the scenario where only a subset of traces in the event log has a label, and (iv) we adopt several context-aware attributes over common execution patterns manifested in the traces. An approach based on sequence patterns and execution time is presented in [22] for identifying failures in business processes where failure patterns are defined to be those sequence patterns that manifest in failure instances and not in normal instances. There are several differences of this approach with the one proposed in this paper. Unlike our approach, [22] assumes that all cases are labeled and supports failure patterns defined only in the form of sequence patterns [23] (On the contrary, our approach supports a wide variety of features). Furthermore, [22] is less robust to noise and does not define/use any means of assessing the significance of one failure pattern over others (e.g., it does not use metrics like the support or confidence).

Common or discriminatory patterns can also be uncovered using *trace alignment* [24], [25] by aligning the traces and identifying differences between traces of different classes. However, this requires manual inspection of the alignment to uncover the discriminating patterns. Inspecting for patterns is cumbersome for large datasets. Therefore, in this paper, *we explore the feasibility of automatically extracting signature patterns from event logs, which can be associated to a particular class.*

In the remainder of this section, we focus on related work in the context of fault diagnosis in alignment with our case study. Event correlation based approaches for failure diagnosis have been proposed in [26], [27], [28]. There are also commercial tools such as HP's OpenView Self-Healing Services [29] and IBM's Trivoli [30] for network management. These methods and tools rely on either an existing rule base (typically derived from the Failure Mode and Effect Analysis (FMEA) [31]) or some known dependency models about the system. Either of these is hard to obtain for complex distributed systems and/or flexible systems such as medical systems. Automated identification of probable causes of performance problems in large server systems was proposed in [32]. This approach relies on the availability of well defined measurements on known metrics relevant to performance problems. Techniques such as these work well when one knows apriori what is to be measured; the analysis then focuses mainly on finding correlations over the measured values. However, event logs from high-tech systems such as X-ray machines capture all events that are triggered on/by/within the system and are typically designed for multiple applications (e.g., understanding system usage, debugging software bugs, etc.). These event logs tend to be fine-granular making the analysis challenging. Such fine-grained event logs first require elaborate preprocessing such as defining abstractions and selecting an appropriate scope for

analysis, which can vary depending on the domain and application. Techniques based on the assumption that deviations exist in component interaction patterns during system/application failures have been proposed in [33], [34], [35]. However, these techniques cannot be applied to event logs that do not capture component interactions explicitly.

VI. Conclusions

In this paper, we explored the feasibility of automatically identifying signature patterns that can discriminate between different classes of behavior. We demonstrated that the proposed framework works well, i.e., it is able to uncover signatures with a high accuracy as was illustrated by a real-life case study on malfunctioning X-ray machines. The resulting signatures remain highly accurate even on unseen instances. This indicates that the suggested framework has the potential to become a powerful tool for the diagnosis of failures in X-ray systems. The proposed framework is generic and can be applied in many other domains ranging from embedded systems to forensics and auditing.

Acknowledgements

The authors are grateful to Philips Healthcare for funding the research in process mining. The authors would also like to thank Fan Liu for her support.

References

[1] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, 2011.

[2] A. Rozinat, I. S. M. de Jong, C. W. Günther, and W. M. P. van der Aalst, "Process Mining Applied to the Test Process of Wafer Scanners in ASML," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 39, no. 4, pp. 474–479, 2009.

[3] A. Oliner and J. Stearley, "What Supercomputers Say: A Study of Five System Logs," in *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2007, pp. 575–584.

[4] R. A. Derrig, "Insurance Fraud," *The Journal of Risk and Insurance*, vol. 69, no. 3, pp. 271–287, 2002.

[5] W. J. Rudman, J. S. Eberhardt, W. Peirce, and S. Hart-Hester, "Healthcare Fraud and Abuse," *Perspectives in Health Information Management*, vol. 6, 2009.

[6] R. J. Bolton and D. J. Hand, "Statistical Fraud Detection: A Review," *Statistical Science*, vol. 17, no. 3, pp. 235–249, 2002.

[7] M. Jans, N. Lybaert, and K. Vanhoof, "Internal Fraud Risk Reduction: Results of a Data Mining Case Study," *International Journal of Accounting Information Systems*, vol. 11, no. 1, pp. 17–41, 2010.

[8] E. Kirkos, C. Spathis, and Y.Manolopoulos, "Data Mining Techniques for the Detection of Fraudulent Financial Statements," *Expert Systems with Applications*, vol. 32, no. 4, pp. 995–1003, 2007.

[9] W. S. Yang and S. Y. Hwang, "A Process Mining Framework for the Detection of Healthcare Fraud and Abuse," *Expert Systems with Applications*, vol. 31, no. 1, pp. 56–68, 2006.

[10] T. Cover and P. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[11] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-dimensional Distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[12] R. P. J. C. Bose and W. M. P. van der Aalst, "Abstractions in Process Mining: A Taxonomy of Patterns," in *Business Process Management*, ser. LNCS, U. Dayal, J. Eder, J. Koehler, and H. Reijers, Eds., vol. 5701. Springer-Verlag, 2009, pp. 159–175.

[13] ——, "Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models," in *BPM 2009 International Workshops, Ulm, Germany, September 7, 2009. Revised Papers*, ser. LNBIP, S. Rinderle-Ma, S. Sadiq, and F. Leymann, Eds., vol. 43, 2009, pp. 170–181.

[14] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2006.

[15] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed., ser. Springer Series in Statistics. Springer-Verlag, Berlin, 2002.

[16] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[17] ——, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[18] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, vol. 1215, 1994, pp. 487–499.

[19] B. Liu, W. Hsu, and Y. Ma, "Integrating Classification and Association Rule Mining," in *Fourth International Conference on Knowledge Discovery and Data Mining (KDD)*. The AAAI Press, 1998, pp. 80–86.

[20] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," in *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 2. Morgan Kaufmann, 1995, pp. 1137–1145.

[21] F. Folino, G. Greco, A. Guzzo, and L. Pontieri, "Mining Usage Scenarios in Business Processes: Outlier-Aware Discovery and Run-Time Prediction," *Data & Knowledge Engineering*, vol. 70, no. 12, pp. 1005Ű–1029, 2011.

[22] G. C. Ruiz and M. Sepulveda, "Discovering Potential Failures in Business Processes Extending Process Mining Techniques," in *XXX International Conference of the Chilean Computer Science Society*, 2011.

[23] P. T. G. Hornix, "Performance Analysis of Business Processes through Process Mining," Master's thesis, Eindhoven University of Technology, 2007.

[24] R. P. J. C. Bose and W. M. P. van der Aalst, "Trace Alignment in Process Mining: Opportunities for Process Diagnostics," in *Proceedings of the 8th International Conference on Business Process Management (BPM)*, ser. LNCS, R. Hull, J. Mendling, and S. Tai, Eds., vol. 6336. Springer-Verlag, 2010, pp. 227–242.

[25] ——, "Process Diagnostics Using Trace Alignment: Opportunities, Issues, and Challenges," *Information Systems*, vol. 37, no. 2, pp. 117–141, 2012.

[26] A. T. Bouloutas, S. Calo, and A. Finkel, "Alarm Correlation and Fault Identification in Communication Networks," *IEEE Transactions on Communications*, vol. 42, no. 234, pp. 523–533, 1994.

[27] L. Mariani and F. Pastore, "Automated Identification of Failure Causes," in *Proceedings of the 20th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE Computer Society, 2008, pp. 117–126.

[28] I. Rouvellou and G. W. Hart, "Automatic Alarm Correlation for Fault Identification," in *Proceedings of the Fourteenth Annual International Conference on Computer Communications (IEEE INFOCOM)*. IEEE Computer Society, 1995, pp. 553–561.

[29] Hewlett-Packard, "HP OpenView Self-Healing Services," 2006, www.managementsoftware.hp.com/services/selfhealing_whitepaper.pdf.

[30] IBM, "IBM-Integrated Service Management Software-Trivoli," www.ibm.com/software/trivoli.

[31] D. H. Stamatis, *Failure Mode and Effect Analysis: FMEA from Theory to Execution*. ASQ Quality Press, 2003.

[32] C. Huang, I. Cohen, J. Symons, and T. Abdelzaher, "Achieving Scalable Automated Diagnosis of Distributed Systems Performance Problems," Hewlett-Packard Development Company, Tech. Rep. HP-2006-160(R.1), 2007.

[33] H. Chen, G. Jiang, C. Ungureanu, and K. Yoshihira, "Online Tracking of Component Interactions for Failure Detection and Localization in Distributed Systems," *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, vol. 37, no. 4, pp. 644–651, 2007.

[34] I. Lee and R. K. Iyer, "Diagnosing Rediscovered Software Problems Using Symptoms," *IEEE Transactions on Software Engineering*, vol. 26, no. 2, pp. 113–127, 2000.

[35] D. Yuan, H. Mai, W. Xiong, L. Tan, Y. Zhou, and S. Pasupathy, "SherLog: Error Diagnosis by Connecting Clues from Run-time Logs," in *Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, J. C. Hoe and V. S. Adve, Eds. ACM, 2010, pp. 143–154.