Computers, Materials & Continua
DOI: 10.32604/cmc.2023.032504
Article

Process Mining Discovery Techniques for Software Architecture Lightweight Evaluation Framework

Mahdi Sahlabadi, Ravie Chandren Muniyandi, Zarina Shukur, Faizan Qamar* and Syed Hussain Ali Kazmi

Centre for Cyber Security, Faculty of Information Science and Technology (FTSM), Universiti Kebangsaan Malaysia (UKM), Bangi, 43600, Selangor, Malaysia
*Corresponding Author: Faizan Qamar. Email: faizanqamar@ukm.edu.my
Received: 20 May 2022; Accepted: 01 September 2022

Abstract: This research recognizes the limitation and challenges of adapting and applying Process Mining as a powerful tool and technique in the Hypothetical Software Architecture (SA) Evaluation Framework with the features and factors of lightweightness. Process mining deals with the large-scale complexity of security and performance analysis, which are the goals of SA evaluation frameworks. As a result of these conjectures, all Process Mining researches in the realm of SA are thoroughly reviewed, and nine challenges for Process Mining Adaption are recognized. Process mining is embedded in the framework and to boost the quality of the SA model for further analysis, the framework nominates architectural discovery algorithms Flower, Alpha, Integer Linear Programming (ILP), Heuristic, and Inductive and compares them *vs.* twelve quality criteria. Finally, the framework's testing on three case studies approves the feasibility of applying process mining to architectural evaluation. The extraction of the SA model is also done by the best model discovery algorithm, which is selected by intensive benchmarking in this research. This research presents case studies of SA in service-oriented, Pipe and Filter, and component-based styles, modeled and simulated by Hierarchical Colored Petri Net techniques based on the cases' documentation. Process mining within this framework deals with the system's log files obtained from SA simulation. Applying process mining is challenging, especially for a SA evaluation framework, as it has not been done yet. The research recognizes the problems of process mining adaption to a hypothetical lightweight SA evaluation framework and addresses these problems during the solution development.

Keywords: Software architecture; process mining; hierarchical colored petri Net; architectural discovery algorithms; model discovery algorithm

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1 Introduction

In industry, Software Architecture (SA) is considered one of the critical success factors in a project. SA indicates a system's components and how these components are communicating. A poor SA can be a major source of errors for scalable and large systems. As a result, SAs are evaluated in the industry to confirm they are robust enough. Although many SA evaluation frameworks have been proposed, a few are used in the industry. Therefore, there is a need for proper tools and techniques to form a framework with these features and factors. Indeed, the lack of proper tools and techniques to integrate and adapt the addressed features and factors in an evaluation method hinders SA Evaluation Framework's success [1]. The current research tries to solve this issue. As a result, Process Mining (PM) is suggested to be applied to the Hypothetical SA Evaluation Framework (HSAEF). HSAEF represents a lightweight evaluation framework [1]. In section two and three, the research reviews PM applications in the software engineering area to identify how PM can be applied to the SA evaluation, then the limitations and challenges are recognized to adapt PM and elicit log files. In section four, the methods, tools, and techniques for the adaption of PM in the HSAEF are discussed, and then the best discovery technique is selected. As a result, five SA models are extracted from in-hand log files and the best SA model is selected for further analysis in HSAEF. Five top discovery techniques are benchmarked on three cases to select the best model. The cases are fit to HSAEF's scope of analysis. In section six and seven, the results are discussed along with the course of action to achieve research aims.

2 Previous Research Methods & Materials

Tools are essential to successful SA evaluation. A powerful tool can help architects show a common blueprint of SA through code and visualization. The embedded tool in the SA evaluation framework supports the evaluation's activities toward the automation of analysis within a framework. The tool and technology inside HSAEF is an intrinsic part that fits the framework's functionalities, and structures [1]. This research suggests PM tools and techniques to ingrate with HSAEF. PM is not merely to discover the processes. It tries to make connections between the process model and event logs, so some techniques are devised to analyze this connection [2,3]. Additional information from different perspectives (Van Der Aalst 2011) may extend the discovered model. PM analyzes process models with the aid of performance-oriented analysis and conformance-oriented analysis [4,5].

3 Challenges of Process Mining Integration and Adaption in Software Engineering

This research walked through the existing literature on process mining adaption in the software system to identify the challenges of PM adaption. The reviewed papers are limited as process mining is new and has not been explored completely. First, the keywords of "process mining" and "software" in the title or abstract of papers in scientific databases (IEEE, ACM, Springer, Google scholar.) were searched. Then some articles not in the realm of software engineering were excluded. According to the systematic literature review, no research has been done in our proposed area to our knowledge so far. Having looked at the studies, PM has been used in software engineering mainly in two application categories, and there are necessary steps for the adaption.

3.1 First Category of PM for Software Development

The aim is to retrieve control-flow and structural aspects of software development cycles from existing data sources and software repositories. As a result, Bug trackers, version control systems, and mail archives are related to tracking software development events [6–13].

3.2 Second Category of PM for Software Structure

It is concerned with reconstructing none concrete models of software systems by their executing traces for analyzing software artifacts to support tasks such as debugging or validating [11,14–24]. In this research, the second category is mostly highlighted, and the research in reverse engineering is considered under the second category.

3.3 Steps for Applying PM on Software

It needs four main steps: event logs extraction and integration, transformation and adding multi-perspective of PM. The mining methods can be applied to discover, monitor, and improve actual software procedures using the software's information [25,26]. These steps initially necessitate the preprocessing of the data from the software repositories for making a log [10,12,18,27].

There are three fundamental queries of "What", "When" and "Who", related to an event of preprocess and extract event logs. Then an event log for a process modelling's discovery algorithm can work based on at least four fields: Case ID (or the Trace ID for the process instance), Timestamp, Actor, and Activity [9]. There are certain questions for integration, transformation and adding multi-perspectives to obtained process models such as "how do various process are working together?", "How" is Process Perspective. It emphasizes the activities control flow. By this controlling, it is intended to expose a good characterization of all potential paths. The response of "Who", Organizational Perspective, focuses on the user field and contains the company's involved users and programmers. Moreover, Case Perspective is the response of "What"; it emphasizes cases properties [28,29]. As a result, the event log should be defined. Nonetheless, obtaining event logs is a challenging job. There are four main challenges to obtaining a log file listed in Table 1.

Table 1: Main challenges to obtain a log file

Challenges	Description
Events correlation	Each case contains groups of events. It sounds simple that requirements can relate to the events, but it is quite challenging. It becomes even more challenging when dealing with heterogeneous or legacy systems. In this case, further efforts should be done to correlate events [30,31].
Events timestamp	Events should be ordered in each case. It is not essential events have timestamps for ordering. Timestamps should be recorded on time, especially when a component has a local clock or is prone to record the time with delay. In addition, timestamp's granularity is important.
Data cleaning (snapshots and scoping)	Actually, an event log is a snapshot indicating long-run processes. Case lifetime may exceed recorded period if it still runs after recording stopped. The solution to this problem is removing incomplete cases.
Granularity	The events should be recorded at the proper level of granularity. Sometimes information systems produce low-level events that are detailed. The level of granularity depends on the Software architect's opinion [32].

In this research, PM is suggested to integrate and adapt to HSAEF as a technique for addressing the targeted quality attributes (security and performance) [33]; however, features and factors were identified by [1]. Despite this progress, the PM's applicability to software engineering has two problems. First, the collected data cannot directly feed the process of SA behavior modeling. The above discussed two problems raise Process Mining Adaption Challenges (PMAC). PMAC challenges are listed in Table 2.

Table 2: PM adaption challenges (PMAC)

PMAC1	Logs record the occurrence of events but do not talk about what could not happen. This means negative examples do not exist [34], assuming the nature of accessible data; the weakness is that the PM algorithms only comprise cases that have occurred; however, it does not include any example of processes that cannot occur. The absence of negative examples is one of the disadvantages of these techniques [10].
PMAC2	There are problems in the preprocessing of event logs contaminated by noise or incomplete [35,36].
PMAC3	As time goes by, processes change (concept drift) [37,38].
PMAC4	It is hard to specify attributes for event logs [35,36].
PMAC5	Choices, loops, and concurrency make a complex structure for the search space, as a result, the log normally consists of all possible behaviors fractions [39].
PMAC6	There is not any significant relationship between the size and behavior of a model. For example, more or fewer behaviors may be generated by a smaller model; however, classical analysis and evaluation methods, in most cases, presume some monotonicity properties [39].
PMAC7	How to balance quality criteria (Generalization, simplicity, fitness and precision) [35].
PMAC8	Need for usability and understandability improvement for non-professionals [40].
PMAC9	The process can be run in various resources which are separated. It is hard to collect events and tailor the processes from a different resource [39].

In HSAEF, SA should be presented in Petri Net format. This presentation has two consequences. First, the simulation is conducted and produced data for PM, and second, the comparison of SAs models in Petri Net format. Technically, it means how the data should be transformed into Log files and then the best PM discovery algorithm which produces Petri Net should be selected.

4 Proposed Approach

4.1 Tools

PM and Petri Net are the essential techniques that are used to implement HSAEF. Petri Net is widely used in many research areas for modeling and simulation. Tools are developed sufficiently to flexibly model complex and large systems tools [1]. Mostly, these tools facilitate programming by mean of the Graphical User Interface (GUI) [41]. 'Colored Petri net (CPN) tools' is capable of modeling Petri Nets along with programming languages. CPN Tools is developed based on Colored Petri Nets and widely used for modeling and analyzing SA [42,43]. CPN Tools can uniquely produce a full state space [44]. Process Mining framework (ProM) [45] is the de facto standard platform for PM in the academic world. It mainly contains more than 600 plug-in boosts ProM capabilities [46,47]. This research uses the comprehensive and extensible benchmarking framework named Comprehensive Benchmarking Framework for Conformance Checking (CoBeFra) [48]. It is integrated with ProM. It also enables us to calculate the Quality Criteria metrics for the process conformance in a comparative, repeatable and consistent way.

4.2 Process Mining Adaption

In this research, the experiments are conducted based on the play-in, play-out, and replay method [49], as shown in Fig. 1, which enables PM techniques to discover, monitor, and improve SA. More detailed information about SA behaviors is provided as more events are recorded. PM bridges between the designed and implemented SA. It includes process discovery (i.e., extracting process models from an event log), conformance checking (i.e., monitoring deviations by comparing model and log) [50], and automated construction of simulation models and model extension.

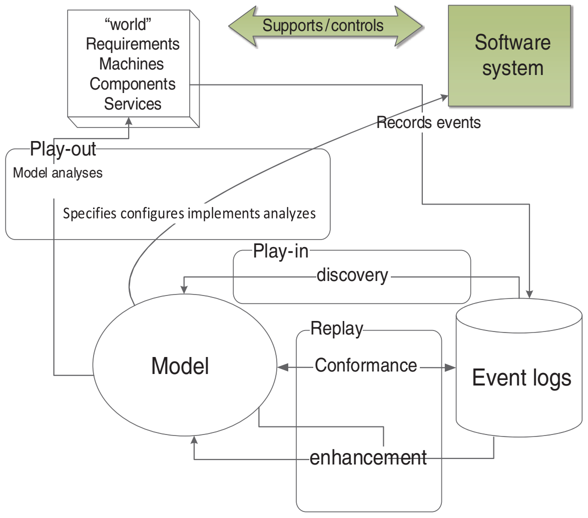


Figure 1: Maps the PM types to play-in, play-out, and replay

PM analysis starts with actual questions. Table 3 states the main use cases of PM and the questions related to these use cases. The use case and its related question are categorized under the type of PM.

Table 3: PM use cases

Use case	Questions	Type of PM	SA's view
Detection of actual SA	What is the SA that actually describes current SA activities?	Discovery, conformance	Late SA analysis [1]
Search bottlenecks in SA	Where are places in the SA, limiting the overall speed of its implementation? What causes these places?	Enhancement	Performance analysis
Detection of deviations in SA	Where the actual process deviates from the expected (ideal) process? Why are there such deviations?	Enhancement	Security analysis

The first type is *discovery*. A discovery method takes an event log and produces a process model without using any a-priori information. For instance, the Alpha algorithm takes an event log and produces a process model (a Petri Net) explaining the behavior recorded in the log. Later, the discovery algorithms are discussed and benchmarked in order to choose the best algorithm for producing the most proper SA model.

The second type is *conformance*. Here, an existing process model is compared with an event log of the same procedure. Conformance checking can be used to check if reality, as recorded in the log, conforms to the model and vice versa. In HSAEF, this type is used to assess the mined SA models to select a "good" SA model. It also is used to check the conformity of the implemented and planned SA.

The third type is *enhancement*. The major idea is to improve and extend an existing process model using information about the actual process recorded in some event logs. Whereas conformance checking measures the alignment between reality and model, this third type of PM targets extending or changing the a-priori model. This type is used to check the performance and security of the SA model.

Orthogonal to the three types of mining, different perspectives can be defined. The control-flow perspective focuses on the ordering of activities goal of mining in order to find a good characterization of all possible paths that can be expressed in Petri Net. The case perspective focuses on the properties of cases. The time perspective is concerned with the frequency and timing of events. It makes it possible to discover bottlenecks, measure service levels, monitor resource utilization, and predict the remaining processing time of running cases.

4.3 Play-In, Play-Out and Replay Method

It uses existing event logs data to model processes that run in implemented SA. Play-In is useful for formally describing the procedures that generate the known information. Petri Nets generate many possible behaviors. The technique of "playing the token game" executes repeatedly and produces traces by means of Petri Nets. Play-out either analyzes models or enacts business processes.

In HSAEF, different scenarios according to the model are simulated for filling the event log by data recorded during the simulation events. This simulation shows the various implementation of the process. Play-Out is used to validate the developed models of processes for compliance with the expected data (sequence of events) with reality. The event log and its relevant process models are both inputted to replay. The event log is replayed based on the process model [51].

The approach attempts to match simulation results with the real model results to find model's deviation of real SA but can also be used to analyze the performance and security of SA. Lastly, replay has as input both event log and process model and necessities for conformance checking, extended the model with frequencies and temporal information, constructing predictive model and operational support.

4.4 Case, Time and Component Perspective

When normally, event logs contain various data. Analysts can exploit it to focus more on specific PM perspectives. In HSAEF, control flow, case, SA component, and time perspectives are highlighted. The control-flow perspective captures the sequence of activities. The case perspective captures data, information and documents produced or required in one case. The component perspective identifies components or roles performing specific activities. The component is like the organization's perspective, referring to components that beget events. The time perspective revolves around the events and timing frequency. It tries to discover bottlenecks, measure service levels, monitors resource utilization, and predict the remaining processing time of running cases. Different perspectives possess common intersections. As it is shown in Fig. 2, in order to analyze the SA model, these five steps should happen.

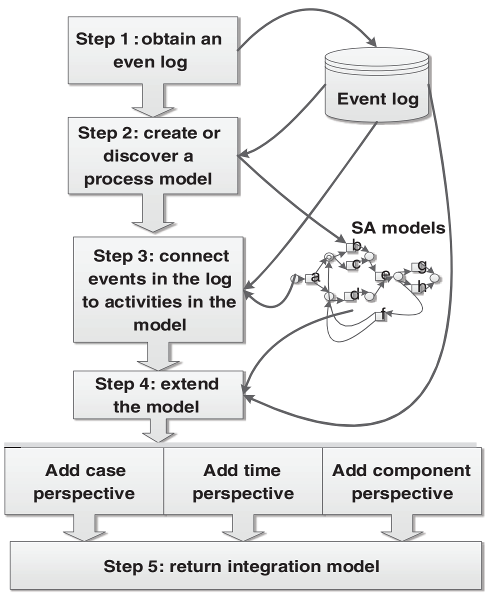

Step 1 : obtain an even log
Event log
Step 2: create or discover a process model
SA models
Step 3: connect events in the log to activities in the model
Step 4: extend the model
Add case perspective
Add time perspective
Add component perspective
Step 5: return integration model

Figure 2: Five steps toward integration of SA mode

5 Experimental Setup

The previous section discusses how the PM can be adapted to HSAEF and obtained loge files. This section explains how the best PM discovery algorithm will be selected. It is crucial for the framework, as the quality of SA models deeply affects the performance and security analysis. Moreover, this section determines the ultimate model of the planned and actual SA model. The discovered SA model quality should be assessed in order to choose the best model. Fig. 3, depicts the process of assessing SA models. As it is shown, there are four main parts. The first part is a simulation that converts UML diagrams and Natural Language (NL) requirements into Hierarchical Colored Petri Nets (HCPN). CPN-tools runs the simulation and produces raw data. Then in the data part, raw data should be preprocessed and exported into Prom to produce log files.

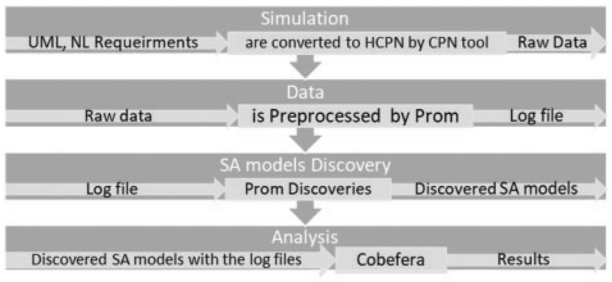


Figure 3: Processes of assessing of mined SA model

The discovery algorithm is necessitated at least four fields: Case/trace ID of the process instance, Actor, Activity, and Timestamp, as mentioned in Section 2. In HSAEF, these fields are mapped to the proper fields based on each case study. The log file will be inputted to the selected discovery algorithms plug-in to produce a set of mined SA models in the discovery part. The models are presented in Petri Net form. In the end all models and the log files will be evaluated by Cobefra. Based on this benchmark's result, the best model will be selected.

5.1 SA Models Discovery

Except for main PM types (discovery, conformance, and enhancement), various perspectives of models also (the organizational/resource perspective, the control-flow, the time and the data perspective) are identified. In HSAEF, the first focus is on the process of discovery. This section aims to assess the quality of discovered mined SA models.

Process discovery is considered the first and most challenging PM type. It is a process model which is constructed based on an event log in order to capture the behavior that has been seen in the log. Discovery techniques use event logs to produce models while there is not any priori information about the models [52]. In HSAEF, the discovery techniques are used to discover the component relations SA and check the validity of some models against the event logs.

Formally, a process discovery algorithm can be defined as a function that maps a log file into a specific process model [53]. A log file contains information for different perspectives, whereas here, the focus is on the discovery models of Petri Nets. Petri Net is graphical and simple, enabling model choices, iteration, and concurrency [53]. A process discovery algorithm is a function γ that maps a log $L \in B(A*)$, in which B is a bag of all possible combinations of the activities of A, onto a marked Petri Net $\gamma(L) = (N, M)$. Ideally, N is all traces in L that correspond to the possible firing sequences of (N, M) and M indicates the current marking. Function γ defines the technique so-called "Play-in" [54].

As the classical approaches of model comparison provide only true/false answers, they are not useful in PM. When two processes are too similar, the classical equivalence may result in the processes which are not equivalent because of some exceptional paths [53]. Therefore, this research focuses on comparing a model to an event log instead of comparing two models. Generally, to evaluate a model, a trade-off should be done among the four Quality Criteria:

5.1.1 Fitness

The mined model can indicate the existing behaviors in the log file.

5.1.2 Precision

The mined model can ignore the behaviors that are not totally in the log file.

5.1.3 Generalization

The mined model can generalize the sample behaviors existing in the log file.

5.1.4 Simplicity

The simplicity of the mined model.

Good fitness means that a model is able to replay most of the traces in the log. Poor precision belongs to an underfitting model, which can replay very different traces in the event log. Overfitting goes for a model specific to the event log traces and not generalized enough to replay various traces. Based on Occam's Razor [55], the "simplest process model" that can explain what is observed in the event log is a good model. It is challenging to balance these quality criteria. The nature completeness and noise are different in the realm of SA and no approach attempts to handle SA by PM yet [16,56]. The challenges of process discovery algorithms facing noise and incompleteness are mitigated as the synthesized data used in HSAEF. Moreover, It is hard to compare the discovery algorithms with the classical data mining challenges [57].

5.2 Discovery Algorithms

Many process discovery techniques have been developed in the last decade. Process discovery algorithms are viewed from two perspectives. First, according to the approach, these algorithms can be applied and second, according to the structures which can be discovered and typical problems can be handled. The research [58] looks at discovery approaches from these two perspectives and divides discovery algorithms into the following items:

5.2.1 Early Algorithmic Approaches

Approaches are mostly immature as they cannot handle the noise and incompleteness in the event logs. Moreover, these approaches have problems in discovering the structures of loops or concurrency. One of the examples of these approaches is the Alpha miner.

5.2.2 Heuristic Dependency Based Approaches

Generally, these techniques have been introduced to improve the functionalities of the discovery to handle the short loops, noise, duplicate tasks and non-free choice constructs. One of the examples of these approaches is the Heuristics miner [46,47].

5.2.3 Genetic Approaches

These approaches can discover a wide spectrum of constructs, but they are so time-consuming and problematic in the large scale of log files [59].

5.2.4 Machine Learning-Based Approaches

These techniques were developed based on machine learning techniques to discover control flow. One of the examples of these approaches is the ILP-based approach [58]. The alpha algorithm is considered a milestone for those discovery algorithms dealing with concurrency. Although the Alpha algorithm provides an insightful view of the PM's realm, it cannot deal with the frequency and noise properly. So, the outcomes of the algorithm are not viewed as world-realistic results [28]. Mostly, discovery algorithms provide models with problems such as dead activities, livelocks, deadlocks, improper termination and inability to terminate [60]. In the proposed framework, the Extreme "flower model" model is used to discover log files as a criterion for an over-general model with high fitness. It allows any behavior (a combination of activities), which causes a highly imprecise process model [34,61,62].

Like this study, some studies reviewed mining techniques and Petri-net discovery algorithms [63–67]. HSAEF defines the activities or events belonging to SAs to synthesize event logs. Three applications pertaining to case studies produced event logs from the simulations to reach this research's aim. The quality of the models is investigated and then the performance issues are examined. Finally, Flower, Alpha, ILP, Heuristic, and Inductive algorithms are selected to discover SA models from log files. These algorithms produce Petri Net models, as it is discussed, they are proper to be applied to the synthesized log.

5.3 *Analysis the "Goodness" of the Mined SA Model*

The Selected algorithms should assess the mined SA models' quality. The evaluation is conducted to compare the quality of mined models based on the log file [68,69]. This study uses the comprehensive and extensible benchmarking framework called CoBeFra. As it is shown in Table 4, the qualities of mined models are evaluated from two angels: comprehensibility and accuracy. These elements can be further decomposed. The comprehensibility is based on the structuredness and simplicity. The simplicity represents controlling the construct's number in the process model. Mostly, model element's number is considered the simplicity metrics.

Table 4: Experimental setup

Discovery algorithms		Flower, alpha, ILP, heuristic, inductive
Quality criteria		Metrics
Accuracy	**Recall:** able to replay even log	Fitness
		Negative event recall
	Precision: not over-fitting the event log	Negative event precision
		Behavioral specificity
	Generalization: not under-fitting the event log	Negative event generalization
		Simple behavioral appropriateness
Comprehensibility	**Simplicity:** Occam's razor	Number of arcs, nodes, transitions or cut vertices
	Structuredness: ease of interpenetration	Structural appropriateness
		Simple behavioral appropriateness

The accuracy belongs to a process model that is composed of the recall or fitness. This is the model's ability to replay the existing behavior in the log file. The precision or appropriateness indicates the ability of the model not to replay the unseen and unwanted behaviors (underfitting prevention) and the generalization, which shows the ability of the model to replay the unseen expected or desired behaviors (overfitting prevention). Table 4, shows the experimental setting to test process discovery algorithms when the original model is known [53].

5.3.1 Fitness Recall

Fitness Recall is the primitive accuracy evaluation since it indicates the number of behaviors that exist in the log file and is reflected in the model. So, it can be considered the most important quality of the mined models. Subsequently, the fitness calculates the compensation tokens which are injected in the Petri Net and remained tokens after the replay's completion.

5.3.2 Behavioral Recall

Behavioral Recall delivers the rate of positive events that are classified correctly in the event log. Each positive event is verified that it is repayable by the process model and sequence replay techniques. These metrics successfully deliver the percentage of positive events parsed by the process model, regardless of remaining or missing tokens.

5.3.3 Precision

Precision implies that a process model does not allow unwanted and unseen behaviors to execute.

5.3.4 Behavioral Specificity

Behavioral Specificity delivers the rate of negative events that are classified correctly in the event log when sequences replay. However, the metric includes natural or artificial negative events. The model replays behaviors. Although these behaviors do not belong to the given event log, they are False positives which can be regarded as negative events.

5.3.5 Generalization

Although the precision of the model is crucial, the model must be generalized over the observed behavior. No event log contains all the behaviors, so the Metrics that express the generalization should penalize overly the precise models. Comprehensibility Metrics expresses the structuredness as the ease of interpretation. It is a difficult dimension to measure. The selected count-based simplicity metrics are the ***number of nodes, arcs, places, transitions, or cut vertices, the weighted transition/place arc degree and the average node arc degree*** [48].

The point of the benchmark of the current research is that the weighted artificial negative events are used for checking the conformance. It leads to accurate results, in the case of a complete event log containing just some possible behaviors [34]. True positive (TP) events are assumed as possible events with regard to the process model and the log file, while false-positive (FP) events are assumed as negative events that are induced by a possible given prefix derived from the model. Weighted behavioral precision, Generalization induces the negative events for an event by taking a window [70].

Simple structural appropriateness is a metric that clarifies the versatility of the task labels in connection with the model (the graph's size) [71]. The induction of negative events explicitly supports incomplete event logs [72,73]. The Artificially Generated Negative Events (AGNEs) enriches the event log with the negative events. The event log consists of the complete set of behavioral patterns, which is interpreted that events can only be missing in a log because they are not permitted by the process [74,75].

5.4 Simulation

CPN Tools produces logs for discovery algorithms of PM in a totally controlled environment, which are suitable for testing PM Algorithms. As discussed, the nominated algorithms are compared to assess the "goodness" of models and selection of the best algorithm. Fig. 4 describes the method of comparison of the algorithms.

In this research, three case studies are selected, and the discovered models of each case are tested. As shown in Fig. 3, the simulation and conversation of SA to Petri net models are out of this research scope, and it will be highlighted in another research for implementation of HSAEF.

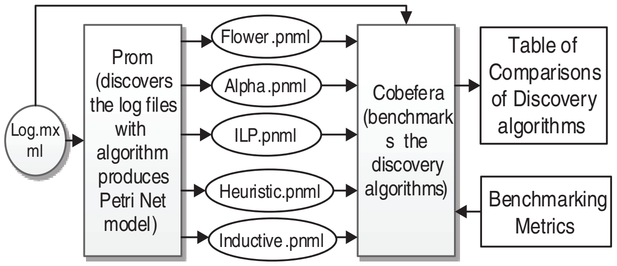

Log.mxml
Prom (discovers the log files with algorithm produces Petri Net model)
Flower.pnml
Alpha.pnml
ILP.pnml
Heuristic.pnml
Inductive.pnml
Cobefera (benchmarks the discovery algorithms)
Table of Comparisons of Discovery algorithms
Benchmarking Metrics

Figure 4: Method of testing of process discovery algorithms

5.5 *Results*

5.5.1 Online Webshop

Online webshops as industrial reference architectures are standards and commonly known. Consequently, the SAs can directly be implemented with minor modifications. Developing the web-based shopping solution with the realization of Service-Oriented Architecture (SOA) is considered a standard reference architecture. These architectures denote the correct ways that subsystems and components should work through. In this sense, the reference architecture is a conceptual architecture or an artifact like SA's diagrams.

5.5.2 Assessing the Discovered SA Models' Results for Online Webshop

Table 5 shows the SA models that are mined by the nominated algorithms. The ILP figure contains approximately 3000 arcs and looks so messy in its small size. The figure is included in this thesis to show the complexity of the discovered SA model.

Table 5: Mined SA models by nominated algorithms for online webshop

Alpha	Flower	ILP[1]	Inductive	Heuristic

5.5.3 Benchmark Result

5.5.4 Personalized Bug Prediction

The repositories of multiple open-source software projects containing datasets of code change histories for several years have been collected. The Datasets involve more than 100 software developers.

The software developers make the different bug patterns based on their experiences and background, from a bug type like wrong literals to wrong if conditionals.

The system supports personalized bug prediction in open-source software by analyzing code pattern changes referring to bugs and bug fixes made by developers over time. The system aims to identify likely locations in a codebase that may contain bugs.

The system extracts the code change histories for a specific period from the various repositories of open-source software projects in the various programming languages such as C, C++, Java, etc. Then the various measures (bugs, bug fixes, location of the bugs, who injects the bug/fixes the bug, time of the bug occurrence, the bug type, the code complexity around the bug, etc.) about the characteristics of the code and the project code are elicited and calculated.

5.5.5 *Assessing the Discovered SA Models' Results Personalized Bug Prediction*

Table 6 shows the SA models that are mined by the nominated algorithms. The ILP and Alpha figure approximately contain 3000 arcs and it looks so messy in the small size. The figure is included in this paper to show the complexity of the discovered SA model.

Table 6: Mined SA models by nominated algorithms for personalized bug prediction

Alpha	Flower	ILP^2	Inductive	Heuristic

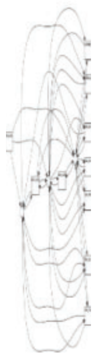
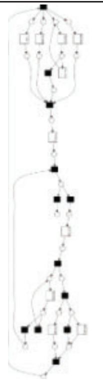
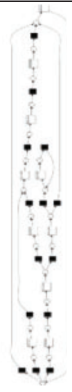
5.5.6 *Benchmark Result*

5.5.7 *Air Conditioner Controlling Through Remote Location*

The system is called "Air conditioner controlling through remote location". This system checks if the air-con system is ON and there is nobody in the room, then it will trigger an alarm to the user's mobile phone to enable the user to switch off the air conditioner through the phone. There are two sensors; one to detect the motion and another to detect the temperature. These sensors are both connected to the Arduino microcontroller board. The microcontroller checks if there is no movement while the temperature is less than a specific degree Celsius, then it will send a message to the user's mobile phone by Global System for Mobile communication (GSM) shield. There is an application on the mobile phone which enables the user to turn the air conditioner off.

The mind models are presented in Table 7, the ILP figures of other case studies contain more than 3000 arcs and look so messy in the small size as a result are not included. The figure is included in this paper to show the complexity of the discovered SA model.

Table 7: Mined SA models by nominated algorithms for air conditioner controller

Alpha	Flower	ILP^2	Inductive	Heuristic
				

5.5.8 Assessing the Discovered SA Models' Results

5.5.9 Benchmark Results

6 Discussion

In this research, the stated questions for these case studies are; "Is it possible to apply PM algorithms for discovering SA?", "What are the discrepancies among process discovery algorithms' output of SA models?", and "which attributes of a SA behavior can be identified by PM algorithms?". The goal of this section is to apply PM algorithms on SA in order to identify the differences between outputs and SA execution. Process discovery algorithm, which can produce Petri Net models, has been selected among the PM algorithms.

As it is shown in Figs. 5–7, the fitness of the Heuristic algorithm is accordingly fairly good (0.6), (0.57), (0.812) rather than other algorithms. It also has almost the lowest value of negative metrics in comparison with other algorithms. It means the model doesn't allow traces that do not belong to the log file be replayed. The simple behavioral and structural appropriateness also is acceptable. Figs. 8–10 confirm that the Heuristic model has the lowest amount of nods and arcs compared to other models, so it is more understandable for stakeholders.

The research assesses the process discovery algorithms based on complex and real event logs. It is concluded that the heuristic miner algorithm is especially suited. The Heuristic Miner (HM) normally handles noise and can indicate the main behaviors residing in the log of activities. It can mine all of the common constructs except the duplicate tasks. The algorithm builds a dependency on the best causal predecessor and successor or a given task. Moreover, this research overcomes the obtaining log file challenges mentioned in Table 1 as:

In HSAEF the scenarios relate to events.

In the proposed framework by adding timed tokens, this problem is alleviated.

In HSAEF, cases with a missing "head" or "tail" are mostly removed by filtering the event log, as starting and finishing activities are known. Information systems may have irrelevant data, which should be omitted from the log file. Domain knowledge of software architects is needed to distinguish required data and to scope data. Obviously, the desired scope depends on both the available data and the questions that need to be answered by the human expert.

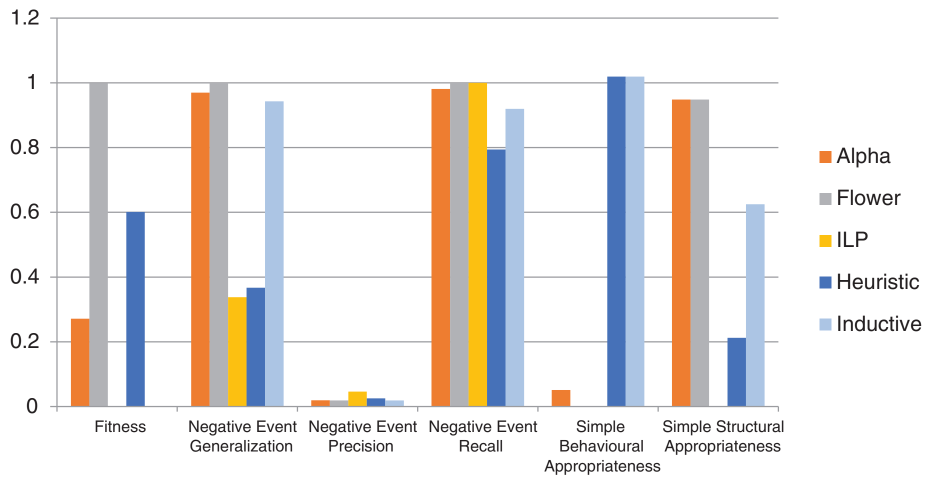


Figure 5: Recall, precision, generalization, simplicity metrics for online

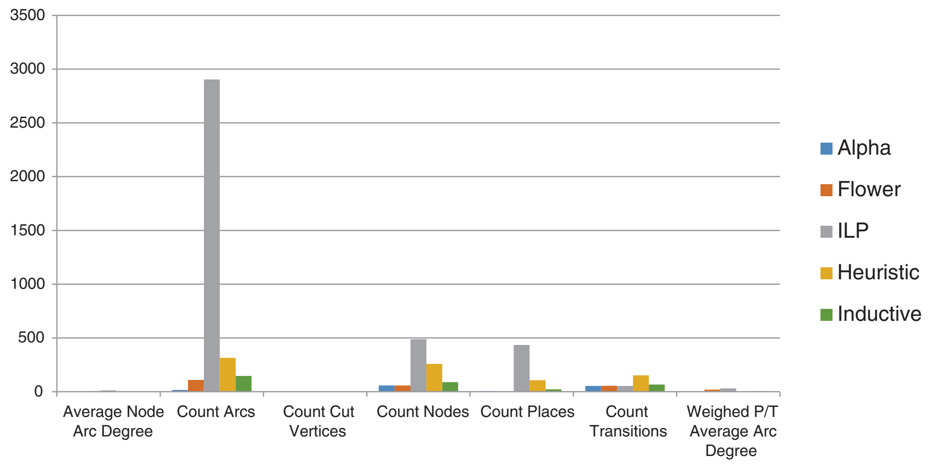


Figure 6: Structuredness metrics for an online webshop

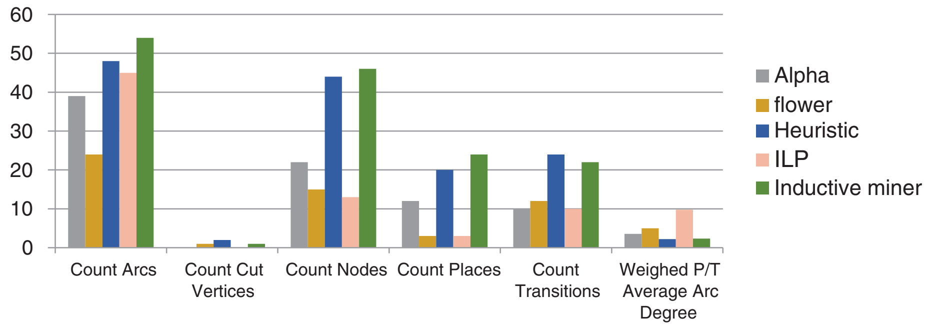


Figure 7: Structuredness for personalized bug prediction

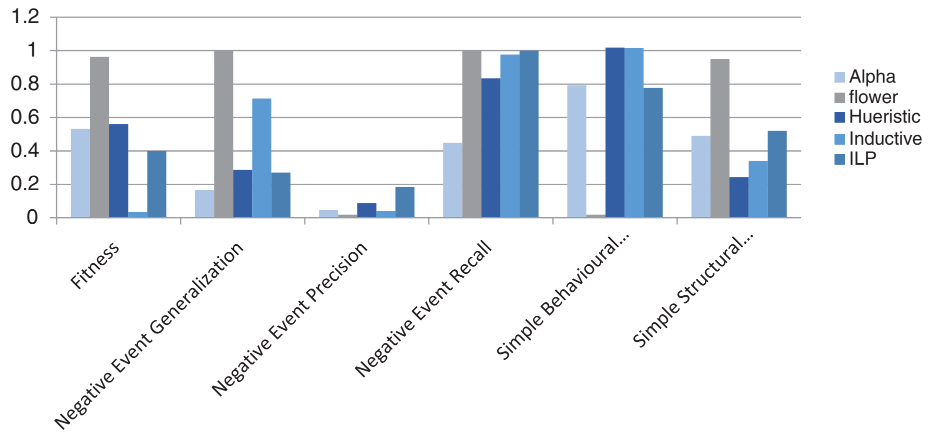


Figure 8: Recall, precision, generalization, simplicity metrics for personalized bug prediction system

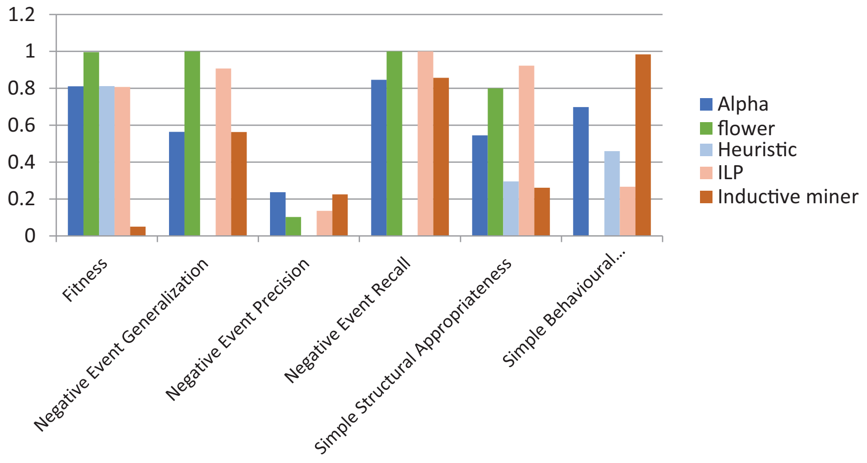


Figure 9: Recall, precision, generalization, simplicity metrics for air conditioner controller

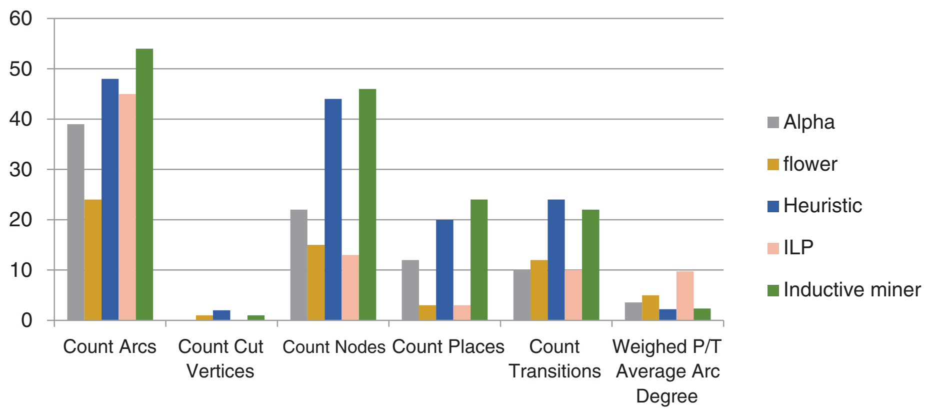


Figure 10: Structuredness for air conditioner controller

In HSAEF, the granularity is determined based on the identified components and the events. The research also addresses PM adaptation challenges mentioned in Table 2 as:

PMAC1: Simulation and the induction of negative events explicitly supports that event logs are incomplete, and the models show better results.

PMAC2: The simulation produces perfect, noise-free and completed log files

PMAC3: The framework can be run periodically.

PMAC4: Use case analysis and add perspectives to ease the process.

PMAC5: Benchmark shows heuristic algorithm can reflect model near to its real one.

PMAC6: We run multiple simulations in different sizes

PMAC7: The five selected algorithms and benchmark criteria alleviate this problem.

PMAC8: Simplicity criteria are considered in the benchmark

PMAC9: Scenario in simulation relate log files.

7 Conclusion

PM should be adapted as a tool in HSAEF. PM produces SA models from the log file, which is the list of the sequence of activities. These activities are associated with an event. The challenges of events' correlation, events' timestamp, data cleaning (snapshots and scoping), scoping and granularity were addressed and mitigated by defining events and logs. Then, the log files were ready to be applied by PM. The strategy of Play-in, Play-out, and Replay was applied to the log files. The fundamental questions of analyzers were answered, and the events were connected to the activities of SA.

In the second step, the discovered SA model's qualities were assessed in order to choose the best SA model. The obtained log files were inputted to the selected discovery algorithms plug-in to produce a set of mined SA models. The models were presented in Petri Net form. In the end, all the models and log files were evaluated by CoBefra. Based on this benchmark's result, the best model was selected. Moreover, PM discovery algorithms and the quality of mined models are assessed and heuristic models are selected. In this paper, we first introduced the challenging problem of process mining to adapt to HSAEF and assess the quality of the mined SAs. We focused on the discovery techniques which produce Petri Net. Hereafter, we presented the benchmark among Alpha, Flower, ILP, and Inductive and Heuristic algorithms. In the experimental session, we showed how Heuristics could deal with the problems based on the given metrics. The results indicate Heuristic algorithm can focus on all behavior in the event log, or only the main behavior, although low frequency behavior and some noise exist in the log.

Acknowledgement: The authors would also like to thank the Universiti Kebangsaan Malaysia (UKM) throughout the research.

Data Availability: The data used to support the findings of this study are available from the corresponding author upon request.

Funding Statement: This paper is supported by Research Grant Number: PP-FTSM-2022.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Sahlabadi, R. C. Muniyandi, Z. Shukur and F. Qamar, "Lightweight software architecture evaluation for industry: A comprehensive review," *MDPI Sensors*, vol. 22, no. 3, pp. 1252, 2022.

[2] J. M. Gama, N. Martin, C. F. Llatas, O. A. Johnson and M. Sepúlvedaet *et al.,* "Process mining for healthcare: Characteristics and challenges," *Journal of Biomedical Informatics Elsevier*, vol. 127, pp. 103994, 2022.

[3] A. Pika, C. Ouyang and A. H. M. Hofstede, "Configurable batch-processing discovery from event logs," *ACM Transactions on Management Information Systems (TMIS)*, vol. 13, no. 3, pp. 1–25, 2022.

[4] J. Theis and H. Darabi, "Decay replay mining to predict next process events," *IEEE Access*, vol. 7, pp. 119787–119803, 2019.

[5] W. M. P. V. Aalst, "Process-aware information systems: Lessons to be learned from process mining," *Transactions on Petri Nets and Other Models of Concurrency II*, vol. 5460, pp. 1–26, 2009.

[6] R. Zhu, Y. Dai, T. Li, Z. Ma and M. Zheng *et al.,* "Automatic real-time mining software process activities from SVN logs using a naive Bayes classifier," *IEEE Access*, vol. 7, pp. 146403–146415, 2019.

[7] J. Caldeira, F. B. E. Abreu, J. Reis and J. Cardoso, "Assessing software development teams' efficiency using process mining," in *Proc. 2019 ICPM*, Aachen, Germany, pp. 65–72, 2019.

[8] A. M. Valle, E. A. Santos and E. R. Loures, "Applying process mining techniques in software process appraisals," *Information and Software Technology*, vol. 87, pp. 19–31, 2017.

[9] P. Juneja, D. Kundra and A. Sureka, "Anvaya: An algorithm and case-study on improving the goodness of software process models generated by mining event-log data in issue tracking systems," in *Proc. 2016 IEEE 40th Annual COMPSAC*, Atlanta, GA, USA, 2016.

[10] B. V. Keith and V. Vega, "Process mining applications in software engineering," in *Proc. Trends and Applications in Software Engineering: CIMPS 2016*, México, USA, pp. 47–56, 2017.

[11] J. S. Stolfa, S. Stolfa, M. S. Kosinar and V. Snasel, "Introduction to integration of the process mining to the knowledge framework for software processes," in *Proc. AECIA 2015*, Villejuif, France, pp. 21–31, 2016.

[12] V. A. A. M. Rubin, A. Alexey, I. A. Lomazova, W. M. P. V. Aalst and M. P. Wil, "Process mining can be applied to software too!," in *Proc. the 8th ACM/IEEE ESEM*, Torino, Italy, no. 57, pp. 1–8, 2014.

[13] J. Caldeira, F. B. Abreu, J. Reis and J. Cardoso, "Assessing software development teams' efficiency using process mining," in *Proc. ICPM, 2019: IEEE*, Aachen, Germany, pp. 65–72, 2019.

[14] M. Cinque, R. D. Corte and A. Pecchia, "Discovering hidden errors from application log traces with process mining," in *Proc. 2019 15th EDCC*, Naples, Italy, pp. 137–140, 2019.

[15] M. Leemans and W. M. P. V. Aalst, "Process mining in software systems: Discovering real-life business transactions and process models from distributed systems," in *2015 Proc. ACM/IEEE 18th MODELS*, Ottawa, ON, Canada, pp. 44–53, 2015.

[16] V. Shah, K. Chetan and S. Rana, "Mining process models and architectural components from test cases," in *Proc. 2015 IEEE Eighth ICSTW*, Graz, Austria, pp. 1–6, 2015.

[17] W. M. P. V. Aalst, "Big software on the run: In vivo software analytics based on process mining," in *Proc. the 2015 ICSSP*, Tallinn, Estonia, pp. 1–5, 2015.

[18] S. A. J. Astromskis, A. Janes and M. Mairegger, "A process mining approach to measure how users interact with software: An industrial case study," in *Proc. the 2015 ICSSP*, NY, USA, pp. 137–141, 2015.

[19] V. A. L. Rubin, I. Lomazova and W. M. P. Aalst, "Agile development with software process mining," in *Proc. the 2014 ICSSP*, Nanjing, China, pp. 70–74, 2014.

[20] C. Liu, B. V. Dongen, N. Assy and W. M. P. V. Aalst, "Component behavior discovery from software execution data," in *Proc. 2016 IEEE SSCI*, Athens, pp. 1–8, 2016.

[21] L. A. D. Cabac and D. Nicolas, "Net components for the integration of process mining into agent-oriented software engineering," in *Transactions on Petri Nets and Other Models of Concurrency I*, Berlin Heidelberg: Springer Berlin Heidelberg, Germany, pp. 86–103, 2008.

[22] R. P. Castillo, B. Weber and M. Piattini, "Process mining through dynamic analysis for modernising legacy systems," *IET Software*, vol. 5, no. 3, pp. 304–319, 2011.

[23] H. Duan, Q. Zeng, H. Wang, S. X. Sun and D. Xu, "Classification and evaluation of timed running schemas for workflow based on process mining," *Journal of Systems and Software*, vol. 82, no. 3, pp. 400–410, 2009.

[24] V. Shynkarenko and O. Zhevaho, "Application of constructive modeling and process mining approaches to the study of source code development in software engineering courses," *Journal of Communications Software and Systems*, vol. 17, no. 4, pp. 342–349, 2021.

[25] V. Rubin, C. W. Günther, W. M. V. D. Aalst, E. Kindler and B. F. V. Dongen *et al.,* "Process mining framework for software processes," in *Proc. the ICSP 2007*, MN, USA, pp. 169–181, 2007.

[26] A. M. Lemos, C. C. Sabino, R. M. F. Lima and C. A. L. Oliveira, "Using process mining in software development process management: A case study," in *Proc. 2011 IEEE SMC*, AK, USA, pp. 1181–1186, 2011.

[27] W. Poncin, A. Serebrenik and M. V. D. Brand, "Process mining software repositories," in *Proc. 2011 15th European CSMR*, Oldenburg, Germany, pp. 5–14, 2011.

[28] R. Saylam and O. K. Sahingoz, "A process mining approach in software development and testing process: A case study," in *Proc. the WCE*, London, UK, vol. 1, 2014.

[29] J. Caldeira and F. B. Abreu, "Software development process mining: Discovery, conformance checking and enhancement," in *Proc. 2016 10th QUATIC*, Lisbon, Portuga, pp. 254–259, 2016.

[30] D. R. Ferreira and D. Gillblad, "Discovering process models from unlabelled event logs," in *Proc. the Int. Conf. on BMP*, Springer, Berlin, Heidelberg, Germany, pp. 143–158, 2009.

[31] N. M. E. Gharib and D. Amyot, "Process mining for cloud-based applications: A systematic literature review," in *Proc. 2019 IEEE 27th Int. REW*, Jeju, Korea (South), pp. 34–43, 2019.

[32] R. P. J. C. Bose and W. M. P. V. D. Aalst, "Abstractions in process mining: A taxonomy of patterns," in *Proc. Int. Conf. on BPM*, Springer, Berlin, Heidelberg, Germany, pp. 159–175, 2009.

[33] D. S. G. Cleiton, A. Meincheim, E. R. F. Junior, M. R. Dallagassa and D. M. V. Sato et al., "Process mining techniques and applications–A systematic mapping study," in *Expert Systems with Application*, vol. 133, pp. 260–295, 2019.

[34] S. K. L. M. V. Broucke, J. D. Weerdt, J. Vanthienen and B. Baesens, "Determining process model precision and generalization with weighted artificial negative events," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1877–1889, 2014.

[35] W. M. P. V. D. Aalst, "Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining," in *Proc. Asia-Pacific Conf. on BPM*, Berlin, Heidelberg, Springer, pp. 1–22, 2013.

[36] M. L. Sebu and H. Ciocarlie, "Applied process mining in software development," in *Proc. 2014 IEEE 9th SACI*, Timisoara, Romania, pp. 55–60, 2014.

[37] R. P. J. C. Bose, W. M. P. V. D. Aalst, I. Žliobaitė and M. Pechenizkiy, "Dealing with concept drifts in process mining," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 151–171, 2014.

[38] M. V. M. Kumar, L. Thomas and B. Annappa, "Phenomenon of concept drift from process mining insight," in *Proc. 2014 IEEE IACC*, Gurgaon, India, pp. 517–522, 2014.

[39] J. C. A. M. Buijs, B. F. V. Dongen and W. M. P. V. D. Aalst, "Towards cross-organizational process mining in collections of process models and their executions," in *Proc. Int. Conf. on BPM*, Springer, Berlin, Heidelberg, Germany, pp. 2–13, 2011.

[40] W. M. P. V. D. Aalst, "Mining additional perspectives," in *Process Mining*, Berlin, Heidelberg, Germany: Springer, pp. 215–240, 2011.

[41] R. Davidrajuh, "Developing a new petri net tool for simulation of discrete event systems," in *Proc. 2008 Second AMS*, Kuala Lumpur, Malaysia, pp. 861–866, 2008.

[42] K. Jensen, L. M. Kristensen and L. Wells, "Coloured petri nets and CPN tools for modelling and validation of concurrent systems," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3, pp. 213–254, 2007.

[43] J. C. Carrasquel, A. Morales and M. E. Villapol, "Prosega/CPN: An extension of CPN tools for automata-based analysis and system verification," *Труды института системного программирования РАН*, vol. 30, pp. 107–128, 2018.

[44] S. Liu, R. Zeng, Z. Sun and X. He, “SAMAT-a tool for software architecture modeling and analysis,” in *Proc. the 24th Int. Conf. on SEKE*, California, USA, pp. 352–358, 2012.
[45] W. M. P. V. D. Aalst, “Process mining software,” in *Process Mining: Data Science in Action*, Berlin Heidelberg: Springer Berlin Heidelberg, Germany, pp. 325–352, 2016.
[46] J. Rudnitckaia and C. Humby, “Process mining: Data science in action,” Semantic Scholar Corpus ID: 44249666, pp. 26–39, 2014.
[47] C. C. Osman and A. M. Ghiran, “When industry 4.0 meets process mining,” *Procedia Computer Science*, vol. 159, pp. 2130–2136, 2019.
[48] S. K. L. M. V. Broucke, J. D. Weerdt, J. Vanthienen and B. Baesens, “A comprehensive benchmarking framework (CoBeFra) for conformance analysis between procedural process models and event logs in ProM,” in *Proc. 2013 IEEE Symposium on CIDM*, Singapore, pp. 254–261, 2013.
[49] W. M. P. V. D. Aalst, “Getting the data,” in *Process Mining Discovery, Conformance and Enhancement of Business Processes*, Berlin, Heidelberg, Germany: Springer, pp. 95–114, 2011.
[50] V. Naderifar, S. Sahran and Z. Shukur, “A review on conformance checking technique for the evaluation of process mining algorithms,” *TEM Journal*, vol. 8, no. 4, pp. 1232–1241, 2019.
[51] W. M. P. V. D. Aalst, “Process mining,” in *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Berlin, Heidelberg, Germany: Springer, 2011.
[52] W. M. P. V. D. Aalst, “Introduction,” in *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Berlin, Heidelberg, Germany: Springer, pp. 1–25, 2011.
[53] W. M. P. V. D. Aalst, “Process discovery: An introduction,” in *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Berlin, Heidelberg, Germany: Springer, pp. 125–156, 2011.
[54] M. Sahlabadi, A. Sahlabadi, R. C. Muniyandi and Z. Shukur, “Evaluation and extracting factual software architecture of distributed system by process mining techniques,” *Asia-Pacific J. Inf. Technol. Multimedia*, vol. 6, no. 2, pp. 77–90, 2017.
[55] W. M. P. V. D. Aalst, “Data mining,” in *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Berlin, Heidelberg, Germany: Springer, pp. 59–91, 2011.
[56] V. Shah, C. Khadke and S. Rana, “Mining process models and architectural components from test cases,” in *Proc. 2015 IEEE Eighth ICSTW*, Graz, Austria, pp. 1–6, 2015.
[57] W. M. P. V. D. Aalst, “Conformance checking,” in *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Berlin, Heidelberg, Germany: Springer, pp. 191–213, 2011.
[58] S. K. L. M. V. Broucke, “Advances in process mining: Artificial negative events and other techniques,” PhD, Ku Leuven Faculteit Economie En Bedrijfswetenschappen, Katholieke Universiteit Leuven, Leuven, Belgium, 2014.
[59] M. Sahlabadi, R. C. Muniyandi and Z. Shukur, “Detecting abnormal behavior in social network websites by using a process mining technique,” *Journal of Computer Science*, vol. 10, no. 3, pp. 393, 2014.
[60] W. M. P. V. D. Aalst, “On the representational bias in process mining,” in *Proc. 2011 IEEE 20th Int. WETICE*, Paris, France, pp. 2–7, 2011.
[61] M. Rahman, S. Halder, M. Uddin and U. K. Acharjee, “An efficient hybrid system for anomaly detection in social networks,” *Cybersecurity*, vol. 4, pp. 1–11, 2021.
[62] W. M. P. V. D. Aalst, “Advanced process discovery techniques,” in *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Berlin, Heidelberg, Germany: Springer, pp. 157–187, 2011.
[63] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa and F. M. Maggi *et al.,* “Automated discovery of process models from event logs: Review and benchmark,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 4, pp. 686–705, 2019.
[64] B. F. V. Dongen, A. K. A. D. Medeiros and L. Wen, “Process mining: Overview and outlook of petri net discovery algorithms,” in *Transactions on Petri Nets and other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems*, Berlin Heidelberg, Germany: Springer, pp. 225–242, 2009.
[65] W. M. P. V. D. Aalstand and B. F. V. Dongen, “Discovering Petri nets from event logs,” in *Transactions on Petri Nets and Other Models of Concurrency VII*, Berlin Heidelberg, Germany: Springer, pp. 372–422, 2013.

[66] A. Rozinat, R. S. Mans, M. Song and W. M. P. V. D. Aalst, "Discovering colored petri nets from event logs," *International Journal on Software Tools for Technology Transfer*, vol. 10, no. 1, pp. 57–74, 2008.
[67] M. Gupta, A. Sureka and S. Padmanabhuni, "Process mining multiple repositories for software defect resolution from control and organizational perspective," in *The Proc. of the 11th Working Conf. on MSR*, Hyderabad, India, pp. 122–131, 2014.
[68] A. Rozinat, A. K. Medeiros, C. W. Günther, A. J. Weijters and W. M. P. V. D. Aalst, "The need for a process mining evaluation framework in research and practice: Position paper," in *The Proc. of the 2007 Int. Conf. on BPM*, Brisbane, Australia, pp. 84–89, 2008.
[69] A. Rozinat, A. K. Medeiros, C. W. Günther, A. J. Weijters and W. M. P. V. D. Aalst, "Towards an evaluation framework for process mining algorithms," in *The Proc. BPM Center Report*, Eindhoven, Netherlands, vol. BPM-07-06, pp. 142, 2007.
[70] N. Tax, X. Lu, N. Sidorova, D. Fahland and W. M. P. V. D. Aalst, "The imprecisions of precision measures in process mining," *Information Processing Letters*, vol. 135, pp. 1–8, 2018.
[71] A. Rozinat and W. M. P. V. D. Aalst, "Conformance checking of processes based on monitoring real behavior," *Information Systems Elsevier*, vol. 33, no. 1, pp. 64–95, 2008.
[72] G. Janssenswillen, T. Jouck, M. Creemers and B. Depaire, "Measuring the quality of models with respect to the underlying system: An empirical study," in *Business Process Management*, Cham: Springer, pp. 73–89, 2016.
[73] S. Goedertier, D. Martens, J. Vanthienen and B. Baesens, "Robust process discovery with artificial negative events," *Journal of Machine Learning Research*, vol. 10, pp. 1305–1340, 2009.
[74] J. Ribeiro, J. Carmona, M. Mısır and M. Sebag, "A recommender system for process discovery," in *Business Process Management*, Cham: Springer, pp. 67–83, 2014.
[75] A. S. A. Polyvyanyy, M. Weidlich, C. D. Ciccio and J. Mendling, "Behavioural quotients for precision and recall in process mining," Technical Report, University of Melbourne, Melbourne, 2018. [Online]. Available: https://minerva-access.unimelb.edu.au/handle/11343/208876?show=full.