*future internet*

MDPI

*Article*

# Statistical Model Checking in Process Mining: A Comprehensive Approach to Analyse Stochastic Processes

**Fawad Ali Mangi [1,2], Guoxin Su [1,*] and Minjie Zhang [1]**

[1] School of Computing and Information Technology, University of Wollongong, Wollongong 2522, Australia; fam366@uowmail.edu.au or fawad.mangi@faculty.muet.edu.pk (F.A.M.); minjie@uow.edu.au (M.Z.)
[2] Department of Computer Systems Engineering, Mehran University of Engineering and Technology Jamshoro, Sindh 76062, Pakistan
* Correspondence: guoxin@uow.edu.au

**Abstract:** The study of business process analysis and optimisation has attracted significant scholarly interest in the recent past, due to its integral role in boosting organisational performance. A specific area of focus within this broader research field is *process mining* (PM) . Its purpose is to extract knowledge and insights from event logs maintained by information systems, thereby discovering process models and identifying process-related issues. On the other hand, *statistical model checking* (SMC) is a verification technique used to analyse and validate properties of stochastic systems that employs statistical methods and random sampling to estimate the likelihood of a property being satisfied. In a seamless business setting, it is essential to validate and verify process models. The objective of this paper is to apply the SMC technique in process mining for the verification and validation of process models with stochastic behaviour and large state space, where probabilistic model checking is not feasible. We propose a novel methodology in this research direction that integrates SMC and PM by formally modelling discovered and replayed process models and apply statistical methods to estimate the results. The methodology facilitates an automated and proficient evaluation of the extent to which a process model aligns with user requirements and assists in selecting the optimal model. We demonstrate the effectiveness of our methodology with a case study of a loan application process performed in a financial institution that deals with loan applications submitted by customers. The case study highlights our methodology's capability to identify the performance constraints of various process models and aid enhancement efforts.

**Keywords:** formal verification; model checking; process discovery; process mining; replay algorithm; statistical model checking



**Citation:** Mangi, F.A.; Su, G.; Zhang, M. Statistical Model Checking in Process Mining: A Comprehensive Approach to Analyse Stochastic Processes. *Future Internet* **2023**, *15*, 378. https://doi.org/10.3390/fi15120378

Academic Editor: Gianluigi Ferrari

Received: 20 October 2023
Revised: 20 November 2023
Accepted: 22 November 2023
Published: 26 November 2023



## 1. Introduction

Modern information systems diligently maintain a record of operational events in the form of an event log. Historically, these system logs served primarily as a reference point to retrospectively trace the sequence of events and identify any potential errors. However, with the rapid progression of technology in today's era, the focus has broadened significantly. The challenge now lies not just in detecting errors but also in deriving valuable insights from these event logs. In the modern era, information systems have become an integral part of various sectors, including business, healthcare, and education, among others. These systems generate a vast number of data, providing a rich source of information for understanding and improving processes. It is necessary to monitor, analyse, and improve a process to keep the business environment running smoothly. However, the complexity and number of these data present significant challenges to extracting meaningful insights.

Process mining [1] has emerged as a promising approach to address these challenges. As an intersection between data mining and business process management, process mining is about discovering, monitoring, analysing, and improving real-world business processes through knowledge extraction from event logs provided by information systems [2].

Real-world examples of event logs include loan applications in banking systems [3], an application of a patient's medical billing report in hospital [4], logs used to detect customer behaviour based on logged clicks [5], and there are more examples.

Real-world processes are often influenced by a number of unpredictable factors, from machine breakdowns to human decision variability. Process-mining techniques might fall short in capturing these peculiarities, leading to models that are overly simplified or not representative of the actual system behaviour. Formal verification has emerged as an effective strategy to verify process correctness. Formal verification is a method used in computer science to verify and/or analyse the properties of finite-state systems, and it is a powerful technique for verifying concurrent and distributed systems. The work presented by [6] introduces BProVe, a verification framework dedicated to *business process model and notation (BPMN)*. This framework was created to address the lack of formal semantics in BPMN, which creates obstacles for the automated verification of relevant properties. The work in [7] proposes an approach, termed model mining, which facilitates the creation of abstract, concise, and functionally structured models from event logs. In [8], a methodology for conducting temporal analysis on mobile service-based business processes is introduced with a concentrated emphasis on the specification and validation of temporal constraints.

Model checking is an often-utilised technique of formal verification that involves the creation of a model of the system, specifying desirable properties in a logical language and then using algorithms to determine whether the model satisfies these properties [9]. According to [10], the model-checking approach model processes as transition systems and express properties as formulas in temporal logic. These properties signify the requirements that a particular process needs to fulfill to be deemed correct. Linear temporal logic (LTL) is one of the languages that allows users to define properties for verification. The objective of model checking is to ascertain whether a process model demonstrates particular desirable behaviour. In some cases where the process model fails to exhibit the necessary behaviour, model-checking techniques provide a counterexample. This assists in identifying the areas of the model that require correction, ensuring that the final model is robust and fit for its intended purpose. The application of model-checking techniques in process mining is a relatively new and promising research area [11]. The integration of these two fields can provide a more robust and comprehensive analysis of operational processes. By applying model-checking techniques to the models discovered in process mining, we can verify whether certain properties hold, detect discrepancies between the intended and actual process, and even predict future process behaviour [12–14].

Historically, the verification of software and hardware systems was largely deterministic. Yet, as systems have become more complex and have begun to account for randomness and uncertainty, deterministic methods have been found lacking in comprehensiveness. The traditional binary output (YES or NO) of model checking may not be enough for real-world processes ranging from financial [15] to automotive [16] and healthcare [17] sectors having stochastic behaviour.

Probabilistic model checking has risen to prominence in this context, providing a means to verify systems that incorporate probabilistic behaviours, with probabilistic models being intended to quantitatively assess system performance and precision [18]. In recent times, probabilistic model checking has established itself as a powerful method for verifying both hardware and software. This technique is recognised for its broad applications, notably in the assessment of reliability, dependability, and performance [19,20]. A noteworthy collaboration was achieved by combining probabilistic model checking and process mining. This was implemented using discrete- and continuous-time Markov chain models, as detailed in [21,22]. The authors proposed a framework to verify whether a discovered model satisfies certain desired properties or constraints and provided answers to questions of how likely, how long, and what factors influence possible paths in a process. Figure 1 illustrates the probabilistic model-checking method for process-mining models.

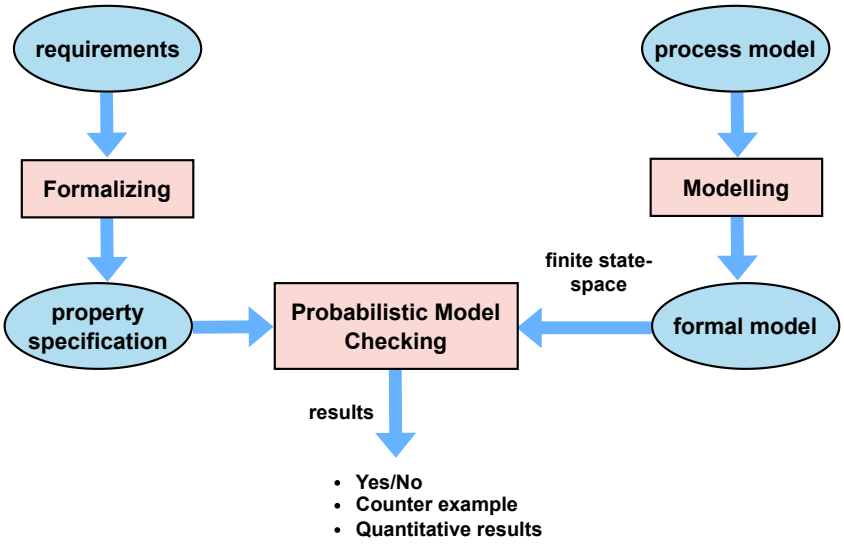


**Figure 1.** Probabilistic model checking for process-mining models.

While probabilistic model checking has made significant advancements in the field of software model checking, showcasing its effectiveness and robustness in approach, the persistent challenge of the state explosion problem continues to require innovative strategies to be overcome. One such solution is the application of statistical model checking (SMC), utilising simulation methods, particularly *discrete-event simulation (DES),* as a mitigation strategy.

SMC represents a methodology rooted in simulation for validating properties relevant to stochastic systems. It serves as an alternative to conventional model-checking techniques, notably the probabilistic ones, which often struggle with systems that exhibit large or infinite state space—a scenario that usually presents challenges for standard methods [23]. Within the scope of SMC, a system is conceptualised as a stochastic process, and the properties are expressed using temporal logic. The fundamental concept supporting SMC is the estimation of the likelihood that a system complies with a specific property. This research article aims to explore the application of statistical-model-checking techniques in process mining.

The methodology presented in this article aims to extend the current process-mining techniques by introducing statistical model checking. In this study, we propose a novel methodology that integrates SMC and PM by formally modelling discovered and replayed process models and applying statistical methods to estimate the results. Our approach offers an efficient, automated evaluation to determine the alignment of a process model with user specifications, aiding users in choosing the most suitable process model. The proposed approach enables the validation of qualitative and quantitative properties of complex systems and/or systems with large state space, where it is difficult to perform probabilistic model checking. The aforementioned process is accomplished by executing several discrete simulations of the system under study, where each simulation trace is individually examined for the property of interest. The final result can be understood as a statistical representation of the given probability, supplemented by a confidence interval. This interval serves to quantify the level of uncertainty embedded in the estimation, thereby providing a comprehensive measure of the probable accuracy of the results.

The novelty of this work lies in the statistical methods utilised, which allow for the analysis of large logs and complex models by using sampling and statistical inference, thus making PM scalable to enterprise-level applications. SMC can provide a formal method to verify business processes against desired properties. This is novel, because traditional PM techniques focus on discovering, monitoring, and improving processes based on historical data without formal verification. By using SMC, the accuracy of discovered process could be improved. SMC-based formal methods in business processes can help in identifying

not just the paths taken in a process, but also those that are possible yet never taken. This provides a complete picture of the process capabilities. SMC can be used to predict future states of a process. Such predictive capability could significantly contribute to PM, allowing organisations to foresee and minimise potential issues before they occur. Figure 2 illustrates the statistical-model-checking method for process-mining models. We believe that our research will contribute to the advancement of process-mining techniques and will provide valuable insights for organisations to improve their processes.

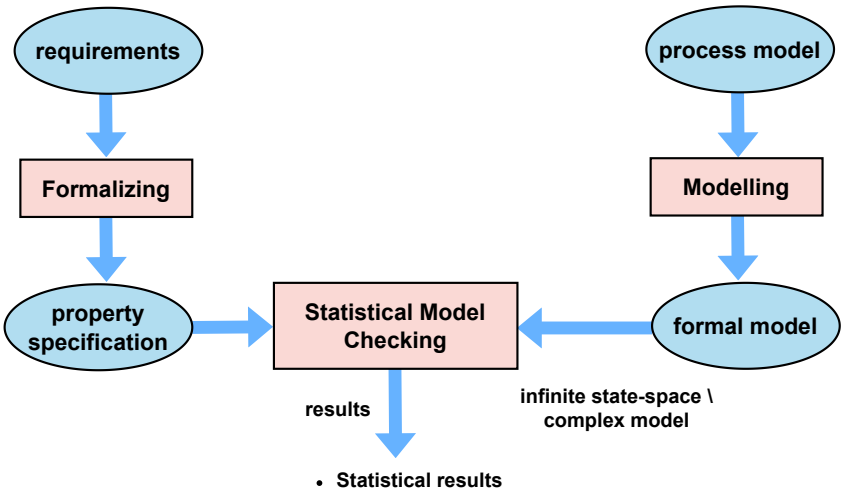


**Figure 2.** Statistical model checking for process-mining models.

The structure of this research article is as follows: Section 2 presents background material related with this research. Section 3 offers a thorough analysis of the relevant work. In Section 4, the proposed framework is presented; it is divided in three stages, and each stage is discussed in detail. The implementation details of a case study and discussion of the results are presented in Section 5, and finally, Section 6 concludes the article, summarising the findings and implications of this research.

## 2. Background

### *2.1. Process Mining*

Process mining serves as a bridge connecting the process-focused aspects of business process management (BPM) with the data-centric elements of machine learning and data mining. The foundation of process mining lies in the analysis of recorded process executions, which are preserved in event logs.

As a field of study, process mining aims to discover, monitor, and refine actual processes by leveraging knowledge extracted from readily accessible event logs in systems [24]. Figure 3 provides a comprehensive overview of the research domain encompassed by process mining. As illustrated in Figure 3, a cloud represents organisations, their business processes, and people. It is the real-world setting where processes happen. The information system supports/controls the operations/activities performed in an organisation (cloud), and it records all events and stores them as event logs, which are detailed records of all the activities that have occurred. A process model is a formal representation of the business processes, and it is derived from the event logs using process-mining techniques, e.g., discovery, conformance, as shown in Figure 3.

As illustrated, process mining establishes a connection between the theoretical behaviour model and the actual observed behaviour.

Process-mining techniques can be categorised into three types: discovery, conformance, and enhancement. There are various types of modelling notations to represent a process model, with *transition systems*, *Petri nets* [25,26], *BPMN* [27], and *causal net* being widely used notations.

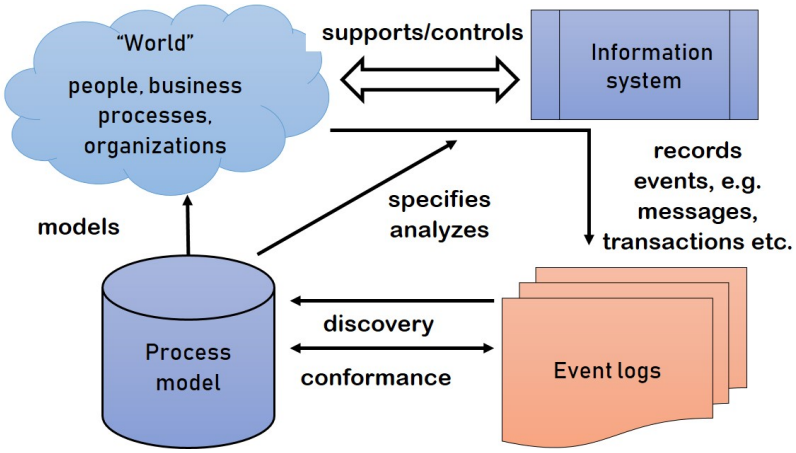


**Figure 3.** Process mining with discovery and conformance [24].

#### 2.1.1. Traces and Event Logs

Let $A$ be a finite set of *activities*. A *trace* $\sigma \in A^*$ is a finite sequence of activities. An *event log* $L$ is a multi-set of traces over a finite set of activities $A$. For example, if $A = \{a, b, c, d, e\}$, then $L = \{\langle a, e, d\rangle, \langle a, c, b, d\rangle^2, \langle a, b, c, d\rangle^3\}$ [24] is an event log including three traces, where traces $\langle a, e, d\rangle$, $\langle a, c, b, d\rangle$, and $\langle a, b, c, d\rangle$ occur once, twice, and three times, respectively.

#### 2.1.2. Petri Net

A Petri net is a formal language with graphical representation consisting of places, transitions, and a flow relation. It is defined as a tuple $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F})$ [24], where $\mathcal{P}$ represents the set of places; $\mathcal{T}$ represents the set of transitions; and $\mathcal{F}$ represents the flow relation, which is a subset of $(\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$. A marked Petri net is denoted by $\mathcal{N}_m = (\mathcal{N}, M_r)$, where $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F})$ and $M_r : \mathcal{P} \to \mathbb{N}$ represents a marking. The marking, also referred to as a *state*, can be understood as a representation that encompasses a collection of places.

### *2.2. Statistical Model Checking*

Statistical model checking is a method that employs simulation techniques to evaluate specific properties defined within a stochastic temporal logic framework. This approach is especially beneficial for assessing systems that are interactive, distributed, and embedded and exhibit stochastic behaviour. Unlike other verification strategies, such as numerical analysis, SMC is distinguished by its capacity to analyse systems with large state space and to accommodate systems with unspecified implementation components. As a versatile and scalable solution, SMC offers a practical alternative for evaluating the properties of stochastic systems, as highlighted in [28–30] and in a comprehensive survey [31].

The conceptual roots of SMC are linked to pioneering work by [32], who initially proposed this technique to explore the properties of probabilistic systems. Their approach integrates Monte Carlo simulation [33] methods, a class of computational algorithms that rely on repeated random sampling to obtain numerical results, and statistical hypothesis testing. In practice, SMC leverages the power of statistical methods, particularly Monte Carlo simulations, to approximate the likelihood that a stochastic system satisfies a particular property. It achieves this by conducting multiple random simulations of the system in question and then applying statistical analysis to estimate the probability that a given property holds. Figure 4 illustrates the general workflow of SMC. As shown in Figure 4, the high-level model is the model of a complex and/or partially uncontrollable system on which we wish to perform SMC. The model generates sample paths, and the number of properties can be utilised to perform verification. As the verification results are statistical in nature, it is required to get an idea of reliable results by running an optimal number of steps, and the analysis stage is responsible for this process by checking the number of paths

and drawing a conclusion. If the number of paths is sufficient to accept the results, then the process ends with drawing a conclusion; otherwise, additional paths are generated from the defined model.

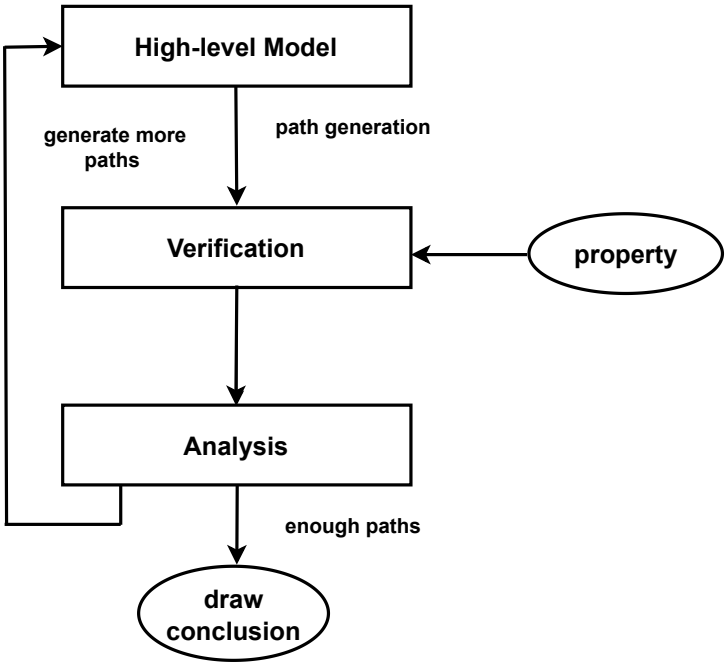


**Figure 4.** SMC general workflow.

In the domain of SMC, statistical tests are instrumental in making informed decisions regarding the system, based on the outputs of the simulation runs. The selection of the test depends on the nature of the verification we intend to achieve—on whether we aim to affirm if the estimated probability surpasses a specific threshold or if it lies within a certain interval. Notably, a number of statistical tests are commonly employed in SMC, such as Wald's sequential probability ratio test (SPRT) [34]; the Chernoff–Hoeffding test [35], which is based on Chernoff–Hoeffding bounds [36]; and the Bayesian test [37].

Meanwhile, an estimator in SMC serves as a protocol for calculating an approximation of a particular quantity based on the data observed. More specifically, in SMC, an estimator's function is to approximate the probability of a path formula or a system property. This is achieved by executing a number of system runs or simulation traces and computes the fraction of runs that comply with the property. Statistical tests and estimators, when incorporated into SMC, enhance its capabilities, making it a practical and robust tool for validating stochastic systems. The power of SMC lies in its ability to effectively deal with complex as well as *black-box*-type systems (a black-box system refers to an operational system where our knowledge is limited to a predetermined set of historical traces acquired during its operational period). Furthermore, the technique provides a method of verification that is statistically valid and exhibits excellent scalability with respect to the dimension of the state space and the complexity of the properties under verification. As a result, SMC has found widespread use in many areas, ranging from computer networks to safety-critical systems, where it provides an indispensable tool for verifying the correctness and performance of stochastic systems.

## 3. Related Work

The advent of process mining has revolutionised the way organisations understand and optimise their operational processes. The advancements in this field have led to significant improvements, enabling experts in their field to understand highly complex processes [38] and non-technical users to easily retrieve relevant information and insights about their processes [39]. By extracting knowledge from event logs, process mining pro-

vides a data-driven lens to analyse and improve business processes. However, the accuracy and reliability of the models generated through process mining are critical for effective decision making. This underscores the need for a robust verification technique to validate these models. Concurrently, the field of formal verification, particularly model checking, has seen substantial advancements. The integration of these two fields, however, is a relatively new area of exploration. This integration can offer a higher degree of confidence in the validity of the models and the insights derived from them.

Formal verification methods are used to prove or disprove the correctness of a system with respect to certain formal specifications or properties. Model checking is an automated method that checks whether a model of a system meets a given specification by exploring all possible states of the system. The work on model checking by [40] serves as a comprehensive overview of the evolution and advancements in model checking, an automatic verification technique for state transition systems, over the past several years. Model checking as a technique has proven to be a valuable tool in the identification and resolution of problems within computer hardware, communication protocols, and software applications. Furthermore, it has begun to find its use in evaluating cyber–physical, biological, and financial systems. However, this approach encounters a significant obstacle, often referred to as the state explosion problem. In simple terms, this issue arises when the sheer number of states within a system becomes too large to be managed effectively [40].

Expanding the scope of model checking, the authors [41] took it into the domain of financial risk assessment. They devised a probabilistic model that provides an assessment of potential risks in financial production. The authors approach is unique, as it relies on computing probabilities to estimate users' behaviours, hence quantifying the likelihood of their actions. The authors recognised for their prior work on process discovery [12] presented a complementary approach by proposing a method to verify whether a specific property holds true for a system, given an event log of the system's behaviour [13]. They accomplish this through the development of a language based on LTL and the use of a standard XML format to store event logs. Furthermore, the authors present an LTL checker that can confirm whether the behaviour observed in an event log complies with the properties defined in the LTL language. The LTL checker was implemented in the process-mining ProM framework [42], and the results can be further analysed using various process-mining tools. This work lays the foundation for the need of verification and model checking of mined models.

In [14], the authors presented an approach that employs temporal logic query checking, which bridges the gap between process discovery and conformance checking. This method facilitates the identification of business rules based on LTL that are consistent with the log by assessing them against a set of user-defined rules. In the study [19,43], the focus was on conformance checking, emphasising the role of alignments, and in [44], they focused on anti-alignments. Such conformance checking provides enhanced diagnostic tools to detect variances between a trace and its associated process model.

The work in [45] advances runtime verification through a comprehensive examination of flexible, constraint-based process models based on LTL on finite traces. In [46], the authors presented a new probabilistic approach to conformance checking by incorporating stochastic process models, considering routing probabilities and frequencies. The work in [47] proposed a runtime verification technique called predictive runtime verification. The unique aspect of this work is that instead of assuming the availability of a system model, the authors describe a runtime verification workflow where the model is learned and incrementally refined using process-mining techniques. The authors in [48] presented an approach for the automatic analysis of business process models discovered through the application of process-mining techniques.

Formal verification techniques such as model checking, however, often encounter a common problem known as the state explosion problem, where the number of states in a model may grow exponentially with the size of the model. Hence, alternative approaches that can handle large models are needed. A unique algorithm that employs Monte Carlo

simulation [33] and hypothesis testing for non-explosive, stochastic discrete-event systems was proposed in [32] and further elaborated in [49]. Monte Carlo simulation—a statistical method—permits the modelling of random variables and their potential outcomes. This approach has significantly improved probabilistic model checking capabilities and highlighted the potential of using simulations to strengthen model-checking methodologies. The authors in [32] presented a model-independent method for verifying properties of discrete-event systems, which often have complex dynamics that are difficult to analyse. The uniqueness of this approach lies in its model independence, making it versatile for any discrete-event system. The sole model-dependent aspect is the establishment of sample execution paths and the probability measure on these path sets. In [50], the algorithm of [32] was extended to statistically verify black-box systems. The paper [50] presents a novel statistical methodology for analysing stochastic systems against specifications specified in a sub-logic of continuous stochastic logic. The system under study behaves as a black-box that is already deployed, from which sample traces can be passively observed but cannot be manipulated. More applications of SMC in stochastic systems can be found in [28–30].

A comprehensive overview of the SMC methodology, its real-world implementations, and its applications in various domains can be found in the literature reviews conducted in [31,51]. These surveys explain how SMC emerges as a formidable strategy for approximating the probability that a system adheres to a certain temporal property. The methodology of SMC involves the examination of a finite collection of system executions, followed by the application of statistical techniques to generalise the findings. The outcomes of this process adhere to a confidence level that can be predefined by the user, thereby allowing for control over the precision of the results. The work emphasises SMC's capacity to alleviate the state-space explosion problem, along with its ability to manage requirements that surpass the expressiveness of traditional temporal logics. The work in [52] introduces a research direction that merges statistical model checking and process mining. The research aims to augment SMC by explaining the reasons behind specific estimates, which can assist in model validation or recommend improvements (enhancement). The integration of SMC and PM results in an approach to white-box behavioural model validation and enhancement. With this methodology, PM can detect discrepancies in the model by examining the simulations generated through SMC. Importantly, since SMC can determine the necessary number of simulations for a given analysis, it produces statistically reliable event logs suitable for PM applications.

The work performed so far in the verification of process models suggests that process models must be validated for accuracy and precision to ensure their effectiveness. This validation is achieved by defining the semantics of the models and applying various logic and formal methods for verification, including model checking. Applications of model checking in process mining are still in their infancy; nevertheless, it is a promising research field and can be extended with the application of statistical model checking in process mining to handle complex systems with stochastic behaviour.

## 4. Proposed Framework

Statistical model checking concentrates on performing the right number of simulations to obtain statistically reliable estimations, such as the probability of success in reaching a goal state. On the other hand, process mining focuses on constructing and/or optimising a system model using logs of its traces. Drawing on the strengths of these two methodologies, as shown in Figures 3 and 4, we propose a framework that integrates SMC and PM. This fusion aids in identifying anomalies and areas of concern within models discovered through PM. Notably, it addresses complex business processes, which are often challenging to model comprehensively. Figure 5 illustrates the proposed framework.

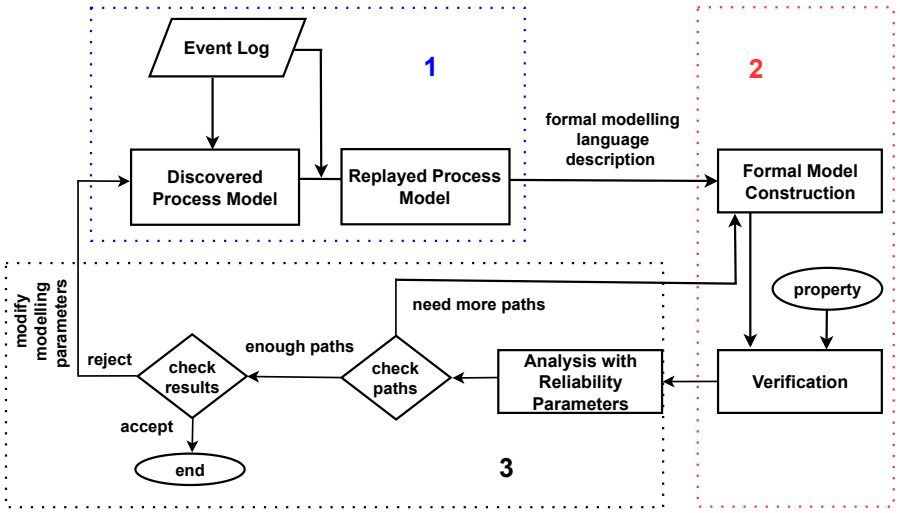


**Figure 5.** Proposed framework.

SMC, with its capacity to manage the state-space explosion problem and its ability to handle complex process behaviours, is a powerful tool for process mining. The application of SMC could greatly aid in understanding, optimising, and controlling complex business processes, thus offering significant value for organisations striving for efficiency and reliability. Furthermore, it could recommend improvements based on the statistically robust results estimated through SMC. The proposed framework is divided into three major parts, as shown below:

1. Process-mining model.
2. Modelling and verification.
3. Analysis.

### *4.1. Process-Mining Model*

In the initial stage, the focus lies in the discovery and replay of the process model. Process discovery is a technique in process mining whereby a process model is constructed from an event log, which is an unstructured sequence of events captured from the execution of a process, while in the replay method, an algorithm operates by replaying event logs in the process model, aiming to reproduce the observed behaviour and evaluate its alignment with the expected process flow. Replay can also provide valuable insights, including the frequency of execution of different paths and performance metrics such as waiting times and execution times for activities. Dotted block 1 in Figure 5 illustrates the initial step of the proposed framework.

#### 4.1.1. Process Discovery

Process discovery entails utilising an event log as an initial data source and generating a model that effectively captures the observed behaviour within the log. The objective of process discovery techniques extends beyond merely constructing models that depict the control flow of activities; they also encompass uncovering additional dimensions, such as revealing the social network connections among the resources involved in executing these activities [53]. The utilisation of process discovery techniques is highly valuable for gaining insights into real-world processes.

*Process discovery algorithm:* Let $I$ denote the set of process discovery algorithms. In a formal sense, a process discovery algorithm can be defined as a function that maps an event log to a process model. Specifically, for each $i \in I$, we have $\mathsf{D}_i : L \mapsto \mathcal{N}$, where $L$ represents the set of event logs and $\mathcal{N}$ represents the resulting process model. The primary objective is to ensure that the discovered process model accurately reflects the observed behaviour within the event log. Figure 6 shows a model of log $L$ discovered using a simple discovery algorithm, *Alpha Miner*. *Alpha Miner*, one of the first and simplest process

discovery algorithms, is utilised for its capability to discover a process model that is easier to understand compared with other discovery algorithms, which may generate complex models.

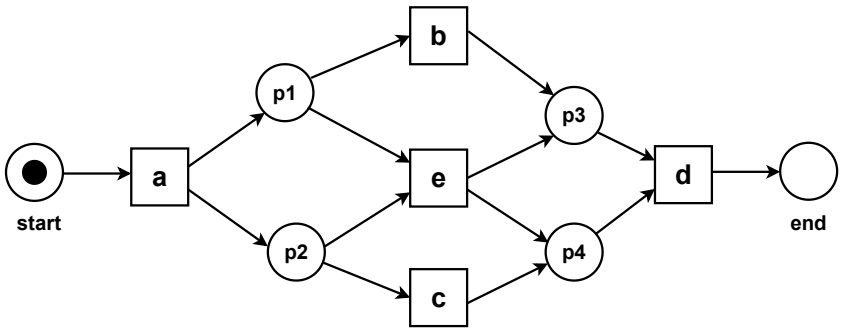


**Figure 6.** Discovered model of log $L$.

#### 4.1.2. Conformance Checking

Conformance checking involves the examination of both a process model and an event log associated with the specific process. The purpose of conformance analysis is to compare the observed behaviour recorded in the log against the behaviour permitted by the model [54,55]. When the observed behaviour deviates from what is allowed by the model or vice versa, it indicates non-conformance between the log and the model.

*Alignment:* It refers to the pairwise comparison or the quantification of the similarity between observed event logs and a reference process model. It measures the level of agreement between the recorded behaviour and the expected behaviour specified by the model. Alignment provides insights into the degree of compliance, accuracy, and fitness of the observed process execution traces with respect to the modelled process.

*Replay algorithm:* A *replay algorithm* refers to a computational procedure that operates on an event log, denoted by $L$, and a process model, represented as $\mathcal{N}$, as its input. The objective of the algorithm is to produce a set of alignments that establish the correspondence between the event log and the process model, denoted by $\mathcal{R}(L, \mathcal{N}) = \Gamma_{L,\mathcal{N}}$. Here, $\Gamma_{L,\mathcal{N}}$ represents a set of alignments that establish a correspondence between the events in the log and the states of a Petri net.

### *4.2. Modelling and Verification*

In this step, the primary focus lies in constructing a model and ensuring its validity for the system under investigation. A model is built to accurately represent the behaviour of the real-world system. The construction of this model typically involves formalising the processes obtained in the first step (Section 4.1), translating them into a formal language that can be used for further analysis. Dotted block 2 in Figure 5 illustrates the modelling and verification step.

In this step, one can build a complete model of a system or can generate paths to perform verification. The formal model is generated with the help of a set of alignments, $\Gamma_{L,\mathcal{N}}$, which results from the replayed process model, as shown in Figure 5. Set of alignments $\Gamma_{L,\mathcal{N}}$ is used to create a transition probability matrix and/or rate matrix depending on the type of model we consider (i.e., discrete-time or continuous-time Markov model). In a formal model of type discrete-time Markov chain (DTMC) $\mathcal{D} = (\mathcal{S}, \bar{s}, \mathcal{L}, P)$ (where $\mathcal{S}$ is a finite set of states, $\bar{s} \in \mathcal{S}$ is the initial state, $\mathcal{L}\colon \mathcal{S} \to 2^{AP}$ is the labelling function, $P\colon \mathcal{S} \times \mathcal{S} \to [0,1]$ is the transition probability matrix), where $\sum_{s' \in S} P(s,s') = 1$ for all $s \in \mathcal{S}$, each element $P(s,s')$ of the transition probability matrix gives the probability of making a transition from state $s$ to $s'$ [56]. And in a formal model of the type continuous-time Markov chain (CTMC) $\mathcal{C} = (S, \bar{s}, L, R)$, $S$ is a finite set of states; $\bar{s} \in S$ is the initial state; $L\colon S \to 2^{AP}$ is the labelling function; and $R\colon S \times S \to \mathbb{R}_{\geq 0}$ is the transition rate matrix [56]. The logic used to express the properties of the formal $DTMC$ model is *probabilistic computation tree logic (PCTL)*, while for $CTMC$, it is *continuous stochastic logic (CSL)*. The syntax and semantics defined below

are for CSL (excluding the $S$ operator), and if we change the time domain from $\mathbb{R}_{\geq 0}$ to $\mathbb{N}$, we obtain PCTL [57]. Let $\Phi$ represent a *state* formula and $\psi$ represent a *path* formula; then,

$$\Phi ::= true \mid ap \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid P_{\sim p}[\psi]$$
$$\psi ::= X\Phi \mid X^{\leq k}\Phi \mid \Phi_1 \; U \; \Phi_2 \mid \Phi_1 \; U^{\leq k} \; \Phi_2$$

where $ap$ is an atomic proposition, $ap \in AP$ (set of atomic propositions), $\sim \in \{<, \leq, \geq, >\}$, $p \in [0,1]$, and $k \in \mathbb{R}_{\geq 0}$. The operators $(\neg, \wedge)$ represent logical $NOT$ and $AND$, respectively, while $X$ ("next"), $X^k$ ("bounded next"), $U$ ("until"), and $U^k$ ("bounded until") are standard operators in temporal logic. Note that we use bounded and unbounded $X$ and $U$ operators. The notion that a state $s$ (or a path $\pi$) satisfies a formula $\Phi$ is denoted by $s \models \Phi$ (or $\pi \models \Phi$) and is defined as follows [57]:

$s \models ap$ iff $ap \in AP(s)$
$s \models \neg\Phi$ iff $s \not\models \Phi$
$s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_1 \wedge s \models \Phi_2$
$s \models P \sim_p (\psi)$ iff Prob $\{\pi \in \text{Path(s)} \mid \pi \models \psi\} \sim p$
$\pi \models X\Phi$ iff $\tau(\pi, 1) < \infty \wedge \pi[1] \models \Phi$
$\pi \models X^{\leq k}\Phi$ iff $\tau(\pi, 1) \leq k \wedge \pi[1] \models \Phi$
$\pi \models \Phi_1 \; U \; \Phi_2$ iff $\exists x \in \mathbb{R}_{\geq 0} \; (\pi(x) \models \Phi_2 \wedge \forall y \in [0, x). \; \pi(y) \models \Phi_1)$
$\pi \models \Phi_1 \; U^{\leq k} \; \Phi_2$ iff $\exists x \in [0, k]. \; (\pi(x) \models \Phi_2 \wedge (\forall y \in [0, x). \; \pi(y) \models \Phi_1))$

This step involves generating a finite number of system simulations and leveraging statistical methods to determine whether the collected samples provide statistical proof of the system either meeting or failing to meet the specified requirements. The key principle here is that the sample runs of a stochastic system are generated in accordance with the system's defined distribution. This allows for the estimation of the probability measure on the system's executions. A formal model of the discovered process, as shown in Figure 6, is presented along with its defined properties below.

```
dtmc
module example_model
  s : [1..5] init 1;
  [] s=1 -> 0.5:(s'=2)+0.33333:(s'=3)+0.16667:(s'=5);
  [] s=2 -> 0.6:(s'=3)+0.4:(s'=4);
  [] s=3 -> 0.4:(s'=2)+0.6:(s'=4);
  [] s=5 -> (s'=4);

endmodule
```

```
const int x;
const int y;

// Checking deadlock state
E [ F "deadlock" ]

// Probability of reaching state x?
P=? [ F s=x ]

// YES or NO verification.
P<0.7 [ (s=x) U (s=y) ]
```

### 4.3. Analysis

The analysis phase within SMC plays an integral role, addressing the results derived from the process and analysing them using a range of key determinants, i.e., *reliability parameters*. A deeper understanding of these reliability parameters provides the ability to closely examine the dependability of the SMC results. Typically, these parameters find associations with the statistical confidence and accuracy of the outcomes obtained from the model-checking procedure. Among the reliability parameters, the concept of *confidence interval* finds prominence, with its elements being confidence level and width. A significant feature within this context is the *statistical test*. An instance of such a test is the sequential probability ratio test (SPRT) [34].

As illustrated in dotted block 3 in Figure 5, the first assessment criterion is the count of paths. This criterion is essential to evaluating the results. If the execution traces or paths are insufficient for making a conclusive decision, additional paths are generated from the established formal model. Conversely, when there are adequate paths executed, the next step involves reviewing the results and proceeding towards making a decision on whether to accept or reject these results. The basis for such decision making, whether to accept or reject the results, resides in the user requirements, serving as the determining factor for the application of statistical-model-checking techniques to process models.

The implication of a user rejecting the acquired results denotes that alterations can be made to the modelling parameters of the process model's discovery and replay. Let us consider a formal model represented by $\mathcal{M}$, and let $\varphi$ represent a temporal logic property. In this scenario, the task of the SMC algorithm is to approximate the value of $\mathbb{P}_{\mathcal{M}}[\varphi]$, where $\mathbb{P}_{\mathcal{M}}[\varphi]$ denotes the probability that $\mathcal{M}$ satisfies $\varphi$. To achieve this, the algorithm generates multiple random paths or samples, denoted by $s_1, s_2, \ldots, s_n$, in $\mathcal{M}$. For every $s_i$, the algorithm checks if $\varphi$ holds. This results in a sequence of Bernoulli random variables $X_i$ defined as $X_i = 1$ if $\varphi$ holds for $s_i$ and $X_i = 0$ otherwise.

The SMC algorithm uses the generated samples to estimate $\mathbb{P}_{\mathcal{M}}[\varphi]$. The estimate is then validated against the user-defined acceptance/rejection criteria that are defined using the reliability parameters. If the estimate falls within acceptable limits, it is accepted. Otherwise, the algorithm iterates, either generating more samples or modifying the model or the property, until a satisfactory estimate is obtained.

## 5. Case Study

### 5.1. Event Log and Process-Mining Algorithms

The implementation of the proposed approach is based on a dataset of a loan application process performed in a financial institution [3]. The data contain all applications filed on an online system in 2016 and their subsequent events until February 2017. There are 31,509 applications filed in total, and for these applications, 42,995 offers were created in total. The events in this dataset represent real-life events captured by the institute's operational processes. The event log contains 26 types of activities, which are divided into 3 types: *Application events*, *Offer events*, and *Workflow events*. To showcase the application of the suggested method, we employed a range of process discovery algorithms, including *Alpha Miner, Discover using Decomposition, Mine for Heuristic Net, and Mine Transition System*. And, for each discovered process model, we implemented various replay algorithms, *Heuristic cost-based fitness with Integer Linear Programming (ILP), A* cost-based fitness with ILP, Best first search simple string distance (SSD) calculation, Dijkstra-based replayer, ILP-based replayer, LP-based replayer, Prefix-based A* cost-based fitness, and Splitting replayer* [58].

### 5.2. Verification

For each of the four process discovery algorithms and each of the eight replay algorithms, we built a DTMC model in the PRISM model checker [59]. Table 1 presents a description of the generated DTMC models grouped by process-mining algorithm. The table is divided into three columns, in which the first column shows process discovery algorithms, the second column is about the number of states for each discovered and replayed

algorithm, and the third column presents the lowest and highest values of transitions for each algorithm. In the second column, we can observe variations in the number of states among discovery algorithms. The discovery algorithm *Mine for Heuristic Net* has 27 states, while the others have 26. The reason is that the *Mine for Heuristic Net* algorithm generates a discovery model that includes all (observable and unobservable) activities. A *(tau)* $\tau$ transition, also known as a silent or invisible transition, represents unobservable activities. It does not correspond to any observable operation or event in the modelled process. It is used to represent internal actions or events that are unobservable or irrelevant to the analysis at hand. For example, in a workflow process, you might have actions like "approve document" or "send email" that are represented by observable transitions. However, there might be internal system checks or data processing steps that are not directly observable from the outside. These are modelled using $\tau$ transitions, indicating that something happens, but it is not necessary to specify what that something is for the purposes of the model. These transitions can change the state of the Petri net without producing any visible outcome.

**Table 1.** Model size of generated DTMC models. For some process discovery algorithms, the transition numbers differ among different replay algorithms.

| PM Algorithm | Num. of States | Num. of Transitions |
|---|---|---|
| Alpha Miner | 26 | [178, 254] |
| Discover using Decomposition | 26 | 178 |
| Mine for Heuristic Net | 27 | [172, 219] |
| Mine Transition System | 26 | [178, 201] |

As mentioned in Section 5.1, there are eight different replay algorithms, which were utilised to generate a formal model. Therefore, the number of transitions varies in each discovery algorithm, depending on the nature of the respective replay algorithm, except for *Discover using Decomposition*. The reason behind this is that the *Discover using Decomposition* algorithm has consistent performance in each replay algorithm. In other words, the *Discover using Decomposition* algorithm maintains consistent behaviour across different scenarios.

The verification procedure is divided into two main steps. Firstly, we undertake *estimation* exercises at *confidence levels of 99% and 95%*. This part of the study fundamentally focuses on the statistical evaluation of the properties under investigation within these confidence intervals. The choices of these specific confidence levels allows us to assess the robustness and reliability of our findings under different degrees of statistical certainty. The 95% confidence level is commonly used in research, as it offers a standard balance between statistical power and confidence. It implies that if the study were repeated multiple times, 95 out of 100 similar studies would produce results within this interval, suggesting a high level of reliability. However, to further strengthen our conclusions and to account for potential variations in the data, we also consider the 99% confidence level. By comparing results at these two levels, we aim to demonstrate the consistency of our findings. If the results remain significant at both the 95% and 99% confidence levels, it indicates a higher degree of certainty in our conclusions. On the other hand, discrepancies between these levels may suggest areas where further investigation is needed or where conclusions are more tentative.

The second part of our verification procedure comprises *hypothesis testing* applied to selected properties. By incorporating these two complementary approaches, we seek to provide a comprehensive evaluation of the properties under investigation. *Estimation* at different confidence levels allows us to assess the reliability of the results across a range of statistical significance levels. *Hypothesis testing*, on the other hand, facilitates the rigorous validation of the underlying assumptions and predictions regarding the properties in question. The proposed framework, therefore, offers a multifaceted and robust evaluation of the model.

5.2.1. Estimation with Confidence Level of 99%

In Tables 2–5, the derived results demonstrate the probabilistic estimates for the examined properties at a confidence level of 99%. In order to obtain reliable results, we conducted a sequence of 100 simulations for each property. Table 2 presents a comprehensive overview of the estimated results specific to each process discovery algorithm, inclusive of the corresponding replay algorithm. The focal property investigated here is expressed as $P = ?\ [F \leq 10\ (s = x)]$. In simpler terms, this property seeks to explore the *likelihood of arriving at state x (where x denotes a target state) within a time frame of 10 units*. The property is particularly useful in scenarios where timely completion or response is critical. It allows stakeholders to assess risks and make informed decisions with high-level confidence based on the likelihood of events within relevant time frames. For example, it can verify requirements such as a task's completion within a specified time frame or the prevention of system failures within a certain period.

**Table 2.** Property $P = ?\ [F \leq 10\ (s = x)]$.

| Process Discovery Algorithm | Replay Algorithm 1 | Replay Algorithm 2 | Replay Algorithm 3 | Replay Algorithm 4 | Replay Algorithm 5 | Replay Algorithm 6 | Replay Algorithm 7 | Replay Algorithm 8 |
|---|---|---|---|---|---|---|---|---|
| Alpha Miner | 0.105 | 0.092 | 0.053 | 0.202 | 0.091 | 0.096 | 0.101 | 0.104 |
| Discover using Decomposition | 0.1 | 0.101 | 0.104 | 0.105 | 0.098 | 0.107 | 0.102 | 0.106 |
| Mine for Heuristic Net | 0.161 | 0.271 | 0.154 | 0.274 | 0.252 | 0.187 | 0.238 | 0.275 |
| Mine Transition System | 0.266 | 0.101 | 0.105 | 0.107 | 0.132 | 0.103 | 0.24 | 0.103 |

The findings pertaining to the particular property expressed as $P = ?\ [\neg\ (s = x)\ U \leq 5\ (s = y)]$ are presented in Table 3. This property represents an exploratory question inquiring about whether the *probability that state x does not occur before state y is encountered within a temporal window of 5 time units*. Essentially, we are investigating the constraint-based temporal behaviour of the model's transitions, imposing a bounded time limit on the *until (U)* operator. The importance of this property lies in its ability to model circumstances where the occurrence of state $y$ takes precedence and should be reached before state $x$. This form of probabilistic property allows for the study of the ordered temporal relationships between different states within a fixed time constraint, offering valuable insights into the temporal dynamics of the system.

**Table 3.** Property $P = ?\ [\neg\ (s = x)\ U \leq 5\ (s = y)]$.

| Process Discovery Algorithm | Replay Algorithm 1 | Replay Algorithm 2 | Replay Algorithm 3 | Replay Algorithm 4 | Replay Algorithm 5 | Replay Algorithm 6 | Replay Algorithm 7 | Replay Algorithm 8 |
|---|---|---|---|---|---|---|---|---|
| Alpha Miner | 0.0133 | 0.0617 | 0.0381 | 0.0392 | 0.0620 | 0.0199 | 0.0128 | 0.0142 |
| Discover using Decomposition | 0.0131 | 0.0118 | 0.0129 | 0.0136 | 0.0140 | 0.0125 | 0.0121 | 0.0133 |
| Mine for Heuristic Net | 0.1354 | 0.1575 | 0.1328 | 0.1200 | 0.1415 | 0.1526 | 0.1261 | 0.1194 |
| Mine Transition System | 0.0192 | 0.0152 | 0.0131 | 0.0126 | 0.0180 | 0.0133 | 0.0139 | 0.0125 |

Table 4 outlines the derived outcomes associated with the verification of the property $R_{cost} = ?\ [F\ (s = x)]$. In essence, this property investigates the *accumulated reward*, represented in this context as a *cost* required to transition to a designated state $x$. It is important to note that state $x$ could represent various scenarios, such as the *target state*, a *deadlock state* that halts progression, or any other significant state within the system model.

**Table 4.** Property $R_{cost} = ?\ [F\ (s = x)]$.

| Process Discovery Algorithm | Replay Algorithm 1 | Replay Algorithm 2 | Replay Algorithm 3 | Replay Algorithm 4 | Replay Algorithm 5 | Replay Algorithm 6 | Replay Algorithm 7 | Replay Algorithm 8 |
|---|---|---|---|---|---|---|---|---|
| Alpha Miner | 97.808 | 51.731 | 54.089 | 54.106 | 51.509 | 98.993 | 98.231 | 97.697 |
| Discover using Decomposition | 99.026 | 99.882 | 98.851 | 98.215 | 98.301 | 98.557 | 99.060 | 98.143 |
| Mine for Heuristic Net | 42.123 | 31.69 | 23 | 29.92 | 30.62 | 31.377 | 47.430 | 37.896 |
| Mine Transition System | 99.467 | 97.626 | 98.595 | 98.781 | 98.636 | 99.596 | 98.181 | 97.89 |

The estimation of such a property is essential, as it facilitates a quantifiable understanding of the resource expenditure needed to transition within the system, thus providing crucial insights into the process model's performance. By quantifying the cost to reach specific states, we are better equipped to evaluate the model's performance under different conditions and make informed decisions. The summary presented in Table 5 offers an analytical exploration of the property $P = ?\ [X\ (s = x)]$, a construct defined to answer the question *what is the likelihood that the subsequent state following the initial state is x?* In this property, we utilise the *NEXT* operator, denoted by $X$, which signifies the next reachable state from the current position in the sequence. The state labelled $x$ is an arbitrary state of interest within the model, subject to the specific requirements or investigative focus of the analysis. This table, therefore, provides a quantitative estimation, offering a calculated probability that the model will transition into specified state $x$ immediately after leaving the initial state. Specific state $x$ is selected based on the research objectives or the specific characteristics of the process under investigation.

**Table 5.** Property $P = ?\ [X\ (s = x)]$.

| Process Discovery Algorithm | Replay Algorithm 1 | Replay Algorithm 2 | Replay Algorithm 3 | Replay Algorithm 4 | Replay Algorithm 5 | Replay Algorithm 6 | Replay Algorithm 7 | Replay Algorithm 8 |
|---|---|---|---|---|---|---|---|---|
| Alpha Miner | 0.6471 | 0.6521 | 0.5545 | 0.6408 | 0.6476 | 0.5541 | 0.6471 | 0.6472 |
| Discover using Decomposition | 0.6502 | 0.6476 | 0.6454 | 0.6447 | 0.6464 | 0.6487 | 0.6462 | 0.6490 |
| Mine for Heuristic Net | 0.6494 | 0.6462 | 0.6524 | 0.6480 | 0.6456 | 0.6470 | 0.6486 | 0.6515 |
| Mine Transition System | 0.6471 | 0.6299 | 0.6255 | 0.6478 | 0.5660 | 0.6055 | 0.6459 | 0.5516 |

#### 5.2.2. Estimation with Confidence Level of 95%

Tables 6–9 present the findings derived from our probabilistic analysis of the properties under investigation. Each of these results provides an estimation with a confidence level of 95%. To ensure a robust set of data, a series of 100 simulations were carried out for each property. Specifically, Table 6 provides a detailed analysis of the projected results, differentiating between a variety of process discovery algorithms and their associated replay algorithms. The focus of our evaluation is the property $P = ?\ [F \leq 10\ (s = x)]$. This property queries the *probability of attaining target state x within a predetermined temporal boundary of 10 units*. A 95% confidence level offers a slightly broader margin for uncertainty compared with the 99% confidence level for the same property, as presented in Table 2. This might be suitable for scenarios where the implications of the outcome are less critical.

**Table 6.** Property $P = ?\ [F \leq 10\ (s = x)]$.

| Process Discovery Algorithm | Replay Algorithm 1 | Replay Algorithm 2 | Replay Algorithm 3 | Replay Algorithm 4 | Replay Algorithm 5 | Replay Algorithm 6 | Replay Algorithm 7 | Replay Algorithm 8 |
|---|---|---|---|---|---|---|---|---|
| Alpha Miner | 0.105 | 0.086 | 0.05 | 0.21 | 0.086 | 0.092 | 0.105 | 0.093 |
| Discover using Decomposition | 0.111 | 0.118 | 0.104 | 0.099 | 0.109 | 0.106 | 0.114 | 0.094 |
| Mine for Heuristic Net | 0.157 | 0.264 | 0.153 | 0.271 | 0.135 | 0.148 | 0.158 | 0.132 |
| Mine Transition System | 0.275 | 0.104 | 0.097 | 0.111 | 0.104 | 0.093 | 0.109 | 0.106 |

The results linked to the specific property expressed as $P = ?\ [\neg\ (s = x)\ U \leq 5\ (s = y)]$ can be found in Table 7. This property represents an investigative query about *the likelihood that condition x does not appear before condition y within a defined period of 5 time units*. The property refers to an examination of model transitions within specific temporal constraints. It captures a restricted duration on the *until* operator, essentially quantifying the time-bound behaviour of these transitions. This property can be used to validate if a system behaves as expected. For instance, in a business process, it can be used to check if a certain undesirable state $(s = x)$ is avoided until a desired state $(s = y)$ is reached within a certain time frame. The comparison between Table 3 and Table 7 highlights the impact of confidence levels on the same property, with Table 3 yielding higher confidence results.

**Table 7.** Property $P = ?\ [\neg\ (s = x)\ U \leq 5\ (s = y)]$.

| Process Discovery Algorithm | Replay Algorithm 1 | Replay Algorithm 2 | Replay Algorithm 3 | Replay Algorithm 4 | Replay Algorithm 5 | Replay Algorithm 6 | Replay Algorithm 7 | Replay Algorithm 8 |
|---|---|---|---|---|---|---|---|---|
| Alpha Miner | 0.013 | 0.063 | 0.031 | 0.039 | 0.052 | 0.010 | 0.017 | 0.014 |
| Discover using Decomposition | 0.015 | 0.015 | 0.016 | 0.015 | 0.021 | 0.013 | 0.016 | 0.018 |
| Mine for Heuristic Net | 0.130 | 0.158 | 0.031 | 0.120 | 0.134 | 0.136 | 0.201 | 0.104 |
| Mine Transition System | 0.022 | 0.016 | 0.014 | 0.014 | 0.015 | 0.017 | 0.016 | 0.015 |

Table 8 presents the results obtained from the verification of the property $R_{\text{cost}} = ?\ [F\ (s = x)]$. Essentially, this property explores the aggregate *reward*, characterised as a *cost* in this scenario, in relation to the transition to a specified state ($x$). Understanding the expected cost until a certain state is reached is crucial for process optimisation. For instance, if $s$ represents a specific state in a process, the property can provide insights into the total cost of reaching state $s$ at a 95% confidence level.

**Table 8.** Property $R_{cost} = ?\ [F\ (s = x)]$.

| Process Discovery Algorithm | Replay Algorithm 1 | Replay Algorithm 2 | Replay Algorithm 3 | Replay Algorithm 4 | Replay Algorithm 5 | Replay Algorithm 6 | Replay Algorithm 7 | Replay Algorithm 8 |
|---|---|---|---|---|---|---|---|---|
| Alpha Miner | 98.041 | 51.128 | 52.511 | 54.189 | 51.151 | 97.429 | 96.365 | 94.495 |
| Discover using Decomposition | 97.740 | 99.061 | 98.222 | 97.266 | 97.708 | 98.691 | 99.396 | 99.029 |
| Mine for Heuristic Net | 62.779 | 31.619 | 22.963 | 29.833 | 32.974 | 32.487 | 47.621 | 38.138 |
| Mine Transition System | 99.435 | 98.210 | 98.915 | 97.469 | 97.295 | 97.486 | 97.155 | 98.469 |

Table 9 provides an in-depth evaluation of the property $P = ?\ [X\ (s = x)]$, a construct designed to estimate *the probability of transitioning into specific state x immediately following the initial state*. This property involves the use of the *NEXT* operator, denoted by $X$, which indicates the subsequent reachable state in the model's sequence. The specific state, denoted by $x$, represents a chosen state in the model. This property can be used to verify the

likelihood of transitioning from one state to another in the very next step. This is particularly useful in processes where immediate transitions are critical, as it aids in understanding the short-term behaviours and immediate consequences of the current state.

**Table 9.** Property $P = ?\ [X\ (s = x)]$.

| Process Discovery Algorithm | Replay Algorithm 1 | Replay Algorithm 2 | Replay Algorithm 3 | Replay Algorithm 4 | Replay Algorithm 5 | Replay Algorithm 6 | Replay Algorithm 7 | Replay Algorithm 8 |
|---|---|---|---|---|---|---|---|---|
| Alpha Miner | 0.651 | 0.635 | 0.555 | 0.642 | 0.651 | 0.651 | 0.645 | 0.643 |
| Discover using Decomposition | 0.660 | 0.656 | 0.649 | 0.644 | 0.643 | 0.635 | 0.649 | 0.655 |
| Mine for Heuristic Net | 0.649 | 0.649 | 0.655 | 0.654 | 0.648 | 0.660 | 0.646 | 0.647 |
| Mine Transition System | 0.659 | 0.653 | 0.657 | 0.653 | 0.654 | 0.656 | 0.644 | 0.651 |

#### 5.2.3. Hypothesis Testing

Hypothesis testing, an essential tool in statistical analysis, can be performed on a set of collected samples. This process involves two distinct methodologies: fixed-size sampling and sequential testing.

In the context of fixed-size sampling, the total number of samples required is determined in advance. This process is systematic, calculating the sample quantity according to risk functions corresponding to error probabilities denoted by $\alpha$ and $\beta$ [60]. It is a predetermined approach where the sample size is set before the commencement of the experiment.

Sequential testing, on the other hand, contrasts with the fixed-size sampling method. In this approach, samples are successively assembled, and statistical analyses are performed concurrently [34]. The process continues iteratively until sufficient information to make a decision with the necessary confidence level is acquired. This method provides flexibility, since the total number of required observations is not set in advance.

The *sequential probability ratio test (SPRT)*, based on *acceptance-sampling* techniques [32], was performed to determine how many samples were needed for the defined property to return a true result within a given bound on the $P$ operator. Table 10 outlines the derived outcomes associated with the verification of the property $P \geq 0.5\ ?\ [\neg\ (s = x)\ U\ \text{“}visited\text{”}]$, which states *what is the probability that state x is not reached until 'visited' is/are reached* (where *'visited'* refers to one or more states defined in the model). This property can be used to determine how many samples are needed to validate whether the system behaves as expected, with a threshold of 50% or more. For instance, in a business process, it can be used to check if a certain undesirable state $(s = x)$ is avoided until number of desired states *visited* is reached with a probability of 50% or more. To obtain reliable results, we conducted a series of 100 simulations for the defined property. Note that the values of $\alpha$ (*type I error*), $\beta$ (*type II error*), and $\delta$ (*half-width of an indifference region*) were set to 0.05.

**Table 10.** Property $P \geq 0.5\ [\neg\ (s = x)\ U\ \text{“}visited\text{”}]$; samples required = ?

| Process Discovery Algorithm | Replay Algorithm 1 | Replay Algorithm 2 | Replay Algorithm 3 | Replay Algorithm 4 | Replay Algorithm 5 | Replay Algorithm 6 | Replay Algorithm 7 | Replay Algorithm 8 |
|---|---|---|---|---|---|---|---|---|
| Alpha Miner | 28 | 29 | 25 | 27 | 29 | 28 | 28 | 28 |
| Discover using Decomposition | 28 | 30 | 27 | 30 | 29 | 27 | 29 | 29 |
| Mine for Heuristic Net | 22 | 29 | 34 | 28 | 28 | 29 | 34 | 24 |
| Mine Transition System | 25 | 31 | 29 | 27 | 29 | 26 | 28 | 26 |

Tables 2–10 illustrate a comprehensive exploration of the application of SMC techniques to process models. The analysis focuses on verifying various properties at different *confidence levels*, specifically, *99% and 95%*, and *hypothesis testing*, specifically, SPRT. Various

statistical outcomes are obtained from different models, each demonstrating varied results for multiple properties, as shown in Tables 2–10. In this context, Tables 2–5 illustrate the estimation results at a 99% confidence level; Tables 6–9 highlight the results at a 95% confidence level; and Table 10 presents the hypothesis-testing results for each process model.

### *5.3. Discussion*

In the research conducted, we utilised statistical-model-checking methods to verify properties associated with a complex system. The outcomes demonstrated significance, highlighting the value of such methods when delving into complex systems based on probability. Initially, our analysis involved the successful implementation of a formal model (*DTMC, in this case*) to represent the system under consideration. This model then underwent verification against various properties expressed using *PCTL*, spanning from simple *reachability* to complex *bounded until* properties. Subsequently, we embarked on an exhaustive statistical model evaluation of these properties. The checking process yielded a series of results, providing valuable insights into the system's behaviour. These insights paved the way for deducing probabilistic metrics significant for informed decisions about system management and control.

During the analysis of properties described in Tables 2–10, which were utilised in the conducted case study, certain observations were made regarding the model's performance, specifically the model discovered through the application of the *Mine for Heuristic Net* technique. The model exhibited better performance in its estimations, irrespective of whether the confidence level was set to *99%* or *95%* . It can be observed from Tables 2–9 that the *Mine for Heuristic Net* discovery algorithm yielded better results in most of the replay algorithms compared with other process discovery algorithms. When considering probabilistic properties where a user desires higher probabilistic values, *Mine for Heuristic Net* provided better results. Similarly, for cost-related properties where the objective is to minimise values, the specified discovery algorithm performed better. During the *hypothesis-testing* phase, all four models under consideration demonstrated varied performance with different replay algorithms. As seen in Table 10, the first replay algorithm required fewer samples for *Mine for Heuristic Net* to return a true result. In contrast, for the second replay algorithm, both *Alpha Miner* and *Mine for Heuristic Net* required a fewer (or the same) number of samples. Additionally, the results vary when moving from left to right in Table 10. Therefore, it is not possible to definitively select one discovery algorithm as superior.

However, it is noteworthy to mention that these results are fundamentally statistical. Despite this statistical nature, the findings indicate that the *Mine for Heuristic Net* model offers certain advantages. Particularly, it appears to be more beneficial in relation to reward- or cost-associated properties, as well as in the estimation of probabilistic properties.

The statistical-model-checking techniques were found to be remarkably effective in handling the inherent randomness of the *DTMC* model. Notably, these techniques offered a practical alternative to traditional, exhaustive model-checking approaches, which are often obstructed by the state-space explosion problem. Moreover, the statistical-model-checking techniques provided both upper and lower bounds to the property probabilities, thereby providing quantifiable uncertainty measures for each result. One critical aspect observed is this study is the trade-off between the accuracy of results and computational cost. While increasing the number of samples in the statistical-model-checking process could enhance the precision of results, it simultaneously led to an increase in the computational load. Therefore, determining an optimal balance between accuracy and computational efficiency is a significant challenge.

In conclusion, the work in this article demonstrates the practicality and efficiency of using statistical-model-checking techniques for verifying the properties of complex models. The results obtained provide a solid foundation for the further use of these methods in more complex and larger probabilistic systems. Our findings also highlight the need for additional research to optimise the balance between result accuracy and computational effort in the model-checking process.

## 6. Conclusions

In this work, we integrate statistical model checking into the domain of process mining. The presented work demonstrates the feasibility and effectiveness of employing SMC to analyse process models extracted from event logs. By applying statistical techniques, we are able to assess complex process behaviours without yielding to the state-space explosion problem, a prevalent issue in traditional model checking. The methodology enables the verification of various properties in a more scalable and efficient manner. Through the combination of SMC and PM, robust tools are provided for businesses to analyse, optimise, and control their processes.

The proposed methodology helps in the continuous monitoring and verification of business processes, ensuring that they meet the desired quality standards. As the methodology sets a new standard for process analysis, it encourages the adoption of forward-thinking approaches to process management. By staying ahead of the curve with such innovative methodologies, businesses can future-proof their processes against evolving operational challenges. The case study presented in this work further solidified the applicability and effectiveness of the proposed approach across different scenarios.

However, this work is not without its limitations and challenges. The tuning of parameters in SMC, such as the confidence level and error margin ($\alpha, \beta$), requires careful consideration and a deep understanding of the underlying processes and statistical principles. Further research may focus on developing more user-friendly methods for non-experts to utilise the power of SMC in process mining. Moreover, while our approach has proven successful in handling various properties and models, certain highly complex or specialised scenarios may require further refinement and extension of our techniques.

**Author Contributions:** Conceptualization, G.S. and F.A.M.; methodology, F.A.M. and G.S.; software, F.A.M.; validation, F.A.M. and G.S.; formal analysis, F.A.M. and G.S.; investigation, F.A.M.; data curation, F.A.M.; writing—original draft preparation, F.A.M.; writing—review and editing, G.S., M.Z. and F.A.M.; visualization, F.A.M. and G.S.; supervision, G.S. and M.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** Data are available at a publicly accessible repository. The data presented in this study are openly available at [3].

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. der Aalst, V.; Mining, W.P. *Discovery, Conformance and Enhancement of Business Processes*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 136.
2. Van Der Aalst, W. Process mining: Overview and opportunities. *ACM Trans. Manag. Inf. Syst. (TMIS)* **2012**, *3*, 1–17. [CrossRef]
3. van Dongen, B. BPI Challenge 2017. 2017. Available online: https://data.4tu.nl/articles/_/12696884/1 (accessed on 10 July 2023).
4. Mannhardt, F.; De Leoni, M.; Reijers, H.A.; Van Der Aalst, W.M. Data-driven process discovery-revealing conditional infrequent behavior from event logs. In *Advanced Information Systems Engineering: 29th International Conference, CAiSE 2017, Essen, Germany, 12–16 June 2017*; Proceedings 29; Springer: Cham, Switzerland, 2017; pp. 545–560.
5. Dees, M.; van Dongen, B. BPI Challenge 2016: Clicks Logged In. 2016. Available online: https://data.4tu.nl/articles/_/12674816/1 (accessed on 15 July 2023).
6. Corradini, F.; Fornari, F.; Polini, A.; Re, B.; Tiezzi, F.; Vandin, A. BProVe: A formal verification framework for business process models. In Proceedings of the 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), Urbana, IL, USA, 30 October–3 November 2017; pp. 217–228.
7. Cerone, A. Model mining: Integrating data analytics, modelling and verification. *J. Intell. Inf. Syst.* **2019**, *52*, 501–532. [CrossRef]
8. Zhao, D.; Gaaloul, W.; Zhang, W.; Zhu, C.; Zhou, Z. Formal Verification of Temporal Constraints for Mobile Service-Based Business Process Models. *IEEE Access* **2018**, *6*, 59843–59852. [CrossRef]
9. Baier, C.; Katoen, J.P. *Principles of Model Checking*; MIT Press: Cambridge, MA, USA, 2008.
10. Clarke, E.M.; Lerda, F. Model checking: Software and beyond. *J. Univ. Comput. Sci.* **2007**, *13*, 639–649.
11. Dumas, M.; La Rosa, M.; Mendling, J.; Reijers, H.A. *Fundamentals of Business Process Management*; Springer: Berlin/Heidelberg, Germany, 2018; Volume 2.

12. Van der Aalst, W.; Weijters, T.; Maruster, L. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 1128–1142. [CrossRef]
13. Van der Aalst, W.M.; de Beer, H.T.; van Dongen, B.F. Process mining and verification of properties: An approach based on temporal logic. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, 31 October–4 November 2005*; Proceedings, Part I; Springer: Berlin/Heidelberg, Germany, 2005; pp. 130–147.
14. Räim, M.; Di Ciccio, C.; Maggi, F.M.; Mecella, M.; Mendling, J. Log-based understanding of business processes through temporal logic query checking. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences: Confederated International Conferences: CoopIS, and ODBASE 2014, Amantea, Italy, 27–31 October 2014*; Proceedings; Springer: Berlin/Heidelberg, Germany, 2014; pp. 75–92.
15. Anderson, B.B.; Hansen, J.V.; Lowry, P.B.; Summers, S.L. Model checking for design and assurance of e-Business processes. *Decis. Support Syst.* **2005**, *39*, 333–344. [CrossRef]
16. Gu, R.; Marinescu, R.; Seceleanu, C.; Lundqvist, K. Formal verification of an autonomous wheel loader by model checking. In Proceedings of the 6th Conference on Formal Methods in Software Engineering, Gothenburg, Sweden, 2 June 2018; pp. 74–83.
17. Munoz-Gama, J.; Martin, N.; Fernandez-Llatas, C.; Johnson, O.A.; Sepúlveda, M.; Helm, E.; Galvez-Yanjari, V.; Rojas, E.; Martinez-Millana, A.; Aloini, D.; et al. Process mining for healthcare: Characteristics and challenges. *J. Biomed. Inform.* **2022**, *127*, 103994. [CrossRef] [PubMed]
18. Katoen, J.P. The probabilistic model checking landscape. In Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, New York, NY, USA, 5–8 July 2016; pp. 31–45.
19. Bergami, G.; Maggi, F.M.; Montali, M.; Peñaloza, R. Probabilistic trace alignment. In Proceedings of the 2021 3rd International Conference on Process Mining (ICPM), Eindhoven, The Netherlands, 31 October–4 November 2021; pp. 9–16.
20. Falcone, Y.; Salaün, G.; Zuo, A. Probabilistic model checking of BPMN processes at runtime. In Proceedings of the International Conference on Integrated Formal Methods, Lugano, Switzerland, 7–10 June 2022; pp. 191–208.
21. Mangi, F.A.; Su, G.; Zhang, M. PM2PMC: A Probabilistic Model Checking Approach in Process Mining. In Proceedings of the 2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET), London, UK, 19–21 May 2023; pp. 1–6.
22. Mangi, F.A.; Su, G.; Zhang, M. Integrating Process Mining with Probabilistic Model Checking via Continuous Time Markov Chains. In Proceedings of the FCS'23, 19th International Conference on Foundations of Computer Science, Las Vegas, NV, USA, 24–27 July 2023; pp. 1–6, (forthcoming).
23. Younes, H.L.S. *Verification and Planning for Stochastic Processes with Asynchronous Events*; Carnegie Mellon University: Pittsburgh, PA, USA, 2004.
24. Van Der Aalst, W.; van der Aalst, W. *Data Science in Action*; Springer: Berlin/Heidelberg, Germany, 2016.
25. Petri, C.A. Kommunikation Mit Automaten. Ph.D. Thesis, University of Bonn, Bonn, Germany, 1962.
26. Reisig, W.; Rozenberg, G. *Lectures on Petri Nets I: Basic Models: Advances in Petri Nets*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1998.
27. OMG, B.P.M. Notation (BPMN) Version 2.0 (2011). 2011; Volume 2. Available online: http://www.omg.org/spec/BPMN/2.0 (accessed on 15 July 2023).
28. Ashok, P.; Křetínskỳ, J.; Weininger, M. PAC statistical model checking for Markov decision processes and stochastic games. In *Computer Aided Verification: 31st International Conference, CAV 2019, New York, NY, USA, 15–18 July 2019*; Proceedings, Part I 31; Springer: Berlin/Heidelberg, Germany, 2019; pp. 497–519.
29. Gros, T.P.; Hermanns, H.; Hoffmann, J.; Klauck, M.; Steinmetz, M. Analyzing neural network behavior through deep statistical model checking. *Int. J. Softw. Tools Technol. Transf.* **2022**, *25*, 407–426. [CrossRef]
30. Agarwal, C.; Guha, S.; Křetínskỳ, J.; Muruganandham, P. PAC Statistical Model Checking of Mean Payoff in Discrete-and Continuous-Time MDP. In Proceedings of the International Conference on Computer Aided Verification, Haifa, Israel, 7–10 August 2022; Springer: Cham, Switzerland, 2022; pp. 3–25.
31. Agha, G.; Palmskog, K. A survey of statistical model checking. *ACM Trans. Model. Comput. Simul. (TOMACS)* **2018**, *28*, 1–39. [CrossRef]
32. Younes, H.L.; Simmons, R.G. Probabilistic verification of discrete event systems using acceptance sampling. In *Computer Aided Verification: 14th International Conference, CAV 2002 Copenhagen, Denmark, 27–31 July 2002*; Proceedings 14; Springer: Berlin/Heidelberg, Germany, 2002; pp. 223–235.
33. Mooney, C.Z. *Monte Carlo Simulation/Christopher Z. Mooney*; Sage university papers series Quantitative applications in the social sciences; No. 07-116; Sage Publications: Thousand Oaks, CA, USA, 1997.
34. Wald, A. Sequential tests of statistical hypotheses. In *Breakthroughs in Statistics: Foundations and Basic Theory*; Springer: New York, NY, USA, 1992; pp. 256–298.
35. Hérault, T.; Lassaigne, R.; Magniette, F.; Peyronnet, S. Approximate probabilistic model checking. In *Verification, Model Checking, and Abstract Interpretation: 5th International Conference, VMCAI 2004 Venice, Italy, 11–13 January 2004*; Proceedings 5; Springer: Berlin/Heidelberg, Germany, 2004; pp. 73–84.
36. Hoeffding, W. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*; Springer Science & Business Media: New York, NY, USA, 1994; pp. 409–426.

37. Jha, S.K.; Clarke, E.M.; Langmead, C.J.; Legay, A.; Platzer, A.; Zuliani, P. A bayesian approach to model checking biological systems. In *Computational Methods in Systems Biology: 7th International Conference, CMSB 2009, Bologna, Italy, 31 August–1 September 2009*; Proceedings 7; Springer: Berlin/Heidelberg, Germany, 2009; pp. 218–234.
38. Werner, M.; Wiese, M.; Maas, A. Embedding process mining into financial statement audits. *Int. J. Account. Inf. Syst.* **2021**, *41*, 100514. [CrossRef]
39. Barbieri, L.; Madeira, E.; Stroeh, K.; van der Aalst, W. A natural language querying interface for process mining. *J. Intell. Inf. Syst.* **2023**, *61*, 113–142. [CrossRef]
40. Clarke, E.M.; Wang, Q. 25 Years of Model Checking. In Proceedings of the Ershov Memorial Conference 2014, St. Petersburg, Russia, 24–27 June 2014; Perspectives of System Informatics; Springer: Berlin/Heidelberg, Germany, 2015; pp. 26–40.
41. Gao, H.; Mao, S.; Huang, W.; Yang, X. Applying probabilistic model checking to financial production risk evaluation and control: A case study of Alibaba's Yu'e Bao. *IEEE Trans. Comput. Soc. Syst.* **2018**, *5*, 785–795. [CrossRef]
42. Van Dongen, B.F.; de Medeiros, A.K.A.; Verbeek, H.; Weijters, A.; van Der Aalst, W.M. The ProM framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets 2005: 26th International Conference, ICATPN 2005, Miami, FL, USA, 20–25 June 2005*; Proceedings 26; Springer: Berlin/Heidelberg, Germany, 2005; pp. 444–454.
43. Leemans, S.J.; Poppe, E.; Wynn, M.T. Directly follows-based process mining: Exploration & a case study. In Proceedings of the 2019 International Conference on Process Mining (ICPM), Aachen, Germany, 24–26 June 2019; pp. 25–32.
44. Boltenhagen, M.; Chatain, T.; Carmona, J. An A-Algorithm for Computing Discounted Anti-Alignments in Process Mining. In Proceedings of the 2021 3rd International Conference on Process Mining (ICPM), Eindhoven, The Netherlands, 31 October–4 November 2021; pp. 25–31.
45. Maggi, F.M.; Westergaard, M.; Montali, M.; van der Aalst, W.M. Runtime verification of LTL-based declarative process models. In *Runtime Verification: Second International Conference, RV 2011, San Francisco, CA, USA, 27–30 September 2011*; Revised Selected Papers 2; Springer: Berlin/Heidelberg, Germany, 2012; pp. 131–146.
46. Leemans, S.J.; van der Aalst, W.M.; Brockhoff, T.; Polyvyanyy, A. Stochastic process mining: Earth movers' stochastic conformance. *Inf. Syst.* **2021**, *102*, 101724. [CrossRef]
47. Ferrando, A.; Delzanno, G. Incrementally predictive runtime verification. *J. Log. Comput.* **2023**, *33*, 796–817. [CrossRef]
48. Zakarija, I.; Škopljanac-Mačina, F.; Blašković, B. Automated simulation and verification of process models discovered by process mining. *Autom. Časopis Za Autom. Mjer. Elektron. Računarstvo I Komun.* **2020**, *61*, 312–324. [CrossRef]
49. Younes, H.L.; Simmons, R.G. Statistical probabilistic model checking with a focus on time-bounded properties. *Inf. Comput.* **2006**, *204*, 1368–1409. [CrossRef]
50. Sen, K.; Viswanathan, M.; Agha, G. Statistical model checking of black-box probabilistic systems. In *Computer Aided Verification: 16th International Conference, CAV 2004, Boston, MA, USA, 13–17 July 2004*; Proceedings 16; Springer: Berlin/Heidelberg, Germany, 2004; pp. 202–215.
51. Legay, A.; Viswanathan, M. Statistical model checking: Challenges and perspectives. *Int. J. Softw. Tools Technol. Transf.* **2015**, *17*, 369–376. [CrossRef]
52. Casaluce, R.; Burattin, A.; Chiaromonte, F.; Vandin, A. Process Mining Meets Statistical Model Checking: Towards a Novel Approach to Model Validation and Enhancement. In Proceedings of the International Conference on Business Process Management, Münster, Germany, 11–16 September 2022; pp. 243–256.
53. Van Der Aalst, W.M.; Pesic, M. DecSerFlow: Towards a truly declarative service flow language. In *Web Services and Formal Methods: Third International Workshop, WS-FM 2006 Vienna, Austria, 8–9 September 2006*; Proceedings 3; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1–23.
54. Rozinat, A.; Van der Aalst, W.M. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.* **2008**, *33*, 64–95. [CrossRef]
55. Adriansyah, A. Aligning Observed and Modeled Behavior. Ph.D. Thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2014. [CrossRef]
56. Kwiatkowska, M.; Norman, G.; Parker, D. Stochastic model checking. In *Formal Methods for Performance Evaluation: 7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2007, Bertinoro, Italy, 28 May–2 June 2007*; Advanced Lectures 7; Springer: Berlin/Heidelberg, Germany, 2007; pp. 220–270.
57. Sen, K.; Viswanathan, M.; Agha, G. On statistical model checking of stochastic systems. In *Computer Aided Verification: 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, 6–10 July 2005*; Proceedings 17; Springer: Berlin/Heidelberg, Germany, 2005; pp. 266–280.
58. Verbeek, H.; Buijs, J.; Van Dongen, B.; van der Aalst, W.M. Prom 6: The process mining toolkit. In Proceedings of the Business Process Management Demonstration Track, Hoboken, NJ, USA, 14–16 September 2010; Volume 615, pp. 34–39.
59. Kwiatkowska, M.; Norman, G.; Parker, D. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11)*, Snowbird, UT, USA, 14–20 July 2011; Volume 6806, pp. 585–591.
60. Wald, A. Statistical decision functions. In *The Annals of Mathematical Statistics*; Edwards Bros.: Kent, UK, 1949; pp. 165–205.