Noname manuscript No.
(will be inserted by the editor)

On the Adoption of Blockchain for Business Process Monitoring

Claudio Di Ciccio · Giovanni Meroni · Pierluigi Plebani

the date of receipt and acceptance should be inserted later

Abstract Being the blockchain and distributed ledger technologies particularly suitable to create trusted environments where participants do not trust each other, Business Process Management represents a proper setting in which these technologies can be adopted. In this direction, current research work primarily focuses on blockchain-oriented business process design, or on execution engines able to enact processes through smart contracts. Conversely, less attention has been paid to study if and how blockchains can be beneficial to business process monitoring. This work aims to fill this gap by (i) providing a reference architecture for enabling the adoption of blockchain technologies in business process monitoring solutions, (ii) defining a set of relevant research challenges derived from this adoption, and (iii) discussing the current approaches to address the aforementioned challenges.

Keywords Blockchain · Distributed Ledger Technology · Business Process Management · Software Architectures · Business Process Monitoring.

1 Introduction

Inter-organizational business processes are the ideal setting in which Distributed Ledger Technologys (DLTs) in general, and blockchain technologies in particular, can be adopted and exploited in their full potential [51]. As a matter of fact, organizations involved in this kind of processes are independent and often do not trust each other. Nevertheless, they collaborate with other

C. Di Ciccio
Sapienza University of Rome, Rome, Italy
E-mail: claudio.diciccio@uniroma1.it

G. Meroni, P. Plebani
Politecnico di Milano, Milan, Italy
E-mail: {giovanni.meroni, pierluigi.plebani}@polimi.it

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

participants in the process to be competitive in the market. Towards this direction, the possibility offered by blockchain technologies and DLTs[1] to provide a shared, immutable, and transparent data storage makes them a valid technology to support such processes. Furthermore, blockchain-based systems do not require the existence of an authoritative third-party actor to control that the shared data have been properly communicated and stored. Such a characteristic allows for the creation of more flexible and dynamic inter-relationships among the involved organizations.

With a specific emphasis on business process monitoring, the blockchain are promising aids as the status of the running process can be shared and trusted among the parties [65]. However, research on the adoption of blockchains for business processes monitoring is still at its early stages. Thus far, most of the attempts have focused on the generation of readily usable data for the application of existing process mining techniques [38,56,21] and for the creation of networks highlighting the most common patterns to exchange information and assets among peers [26,65,32]. A comprehensive analysis of the aspects of blockchains that may favour or encumber process monitoring is, to the best of our knowledge, still missing.

In this prospective article, we pursue the objective of clarifying to what extent a blockchain can be beneficial for business process monitoring and which could be the critical issues to be faced. To this end, after a brief summary about the needs concerning the business process monitoring and the main concepts at the core of blockchains, the paper provides the following three key contributions:

- We analyse the what, why and how of business process monitoring, considering the benefits that the usage of blockchains can bring. In light of this assesment, we identify five requirements that a blockchain-based process monitoring platform should meet.
- We propose a reference architecture to implement a blockchain-based process monitoring solution that addresses the aforementioned requirements. Our architecture takes inspiration from the work of Pourmirza et al. [63] on reference architectures of business process management systems, and is organized along three main layers: (i) an off-chain monitoring layer, where the monitoring data are collected by the process participants; (ii) an on-chain trust layer, where the monitoring data are assessed and become trusted; (iii) an auditing layer, in which trusted monitoring data are used by auditors to acquire information on the status of the process.
- We identify a set of research challenges that impact on the design and realization of blockchain-based process monitoring platforms. These challenges are classified on the aspect of the reference architecture they influence, that is, smart contracts, oracles, data management, and interoperability among platforms.

[1] For the sake of simplicity, we will hereafter use the term *blockchain* to encompass blockchain technologies and DLTs, unless a clear distinction is required.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

– We discuss how some of the existing techniques in the literature can be used to mitigate the research challenges we identified. For each technique, we provide a brief description of how it operates, the characteristics that a blockchain platform should guarantee for it to be applicable, and the limitations it has.

The above contributions revise and extend the idea paper presented in the context of the "BPM meets data" focus theme at the 21st Business Process Modeling, Development and Support (BPMDS) working conference (BPMDS 2020) [20]. This paper is intended to support researchers and practitioners who want to explore and develop solutions for business process monitoring based on blockchain.

The remainder of this paper is as follows. Section 2 describes the concepts on which blockchain platforms are based, and outlines the characteristics to classify blockchain implementations that are relevant for process monitoring. Section 3 provides a motivating scenario, which will be used as a running example throughout this paper. Section 4 describes the fundamental elements of process monitoring and how blockchain can be beneficial for them. Section 5 outlines the reference architecture we propose for blockchain-based process monitoring. Section 6 presents the research challenges that must be faced when designing a blockchain-based monitoring platform. Section 7 discusses how existing research work on blockchain can mitigate some of those challenges. Section 8 illustrates the main research outcomes obtained thus far for the process-oriented analysis of blockchain data. Section 9 discusses the threats to the validity of this work, and the countermeasures we applied to minimize them. Finally, Section 10 concludes the paper and draws future work directions.

2 Elements of blockchains

A blockchain is a protocol for the distributed storage of a tamper-proof sequence of transactions, maintained and verified by the nodes participating in the network. The basic unit of computation is a *transaction*, which records a transfer of value (digital assets, cryptocurrencies, information bits, etc.) between two accounts. The sender cryptographically signs the transaction to provide evidence that it is not counterfeit. Transactions are stored in an append-only list structure – hence the name of DLT. Blockchains – which are members of the family of DLTs – such as Bitcoin[2] [58] and Ethereum[3] [77] collate transactions into so-called blocks. Blocks are thus used as the messages to be broadcast to every node. The order among blocks (and, a fortiori, the transactions therein) is kept by a hash-based backward linking: every block keeps the digest of a hashing function applied to the previous block. All together, the links generate a chain-like structure: hence the name blockchain which refers

[2] https://bitcoin.org/. Accessed on April 13, 2022.
[3] https://ethereum.org/. Accessed on April 13, 2022.

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/s10270-021-00959-x

to the data structure adopted to store the transactions in the ledgers. Locally to a node, transactions are subject to a total ordering relation: the evolution of the state of the parties' accounts depend on the sequence of operations recorded in the ledger. Blocks are, in fact, a measure of time as their addition to the chain determines the passage to the next global system state. To pay back the effort, an economic incentive is proposed that distributes so-called cryptocurrencies to the nodes that contribute to the maintenance and reliability of the network with operations such as the validation of transactions and mining of blocks. Nodes participating in the network guarantee that transactions and blocks are valid and thus prevent the data structure to be tampered with. Also, the replication of the ledger makes it possible to have the stored information always available locally to every node. However, the ledger may differ from node to node: the nodes reach *eventual consensus* on the correct sequence in the ledger. Temporary divergences between the local images of the ledger are called *forks*.

An array of different implementations exists, providing readily usable blockchain platforms. Starting from the taxonomy presented in [72], we analyzed existing and ongoing research work aiming at addressing the problem of process monitoring with the blockchain, which will be discussed in detail in Section 8. As a result, we identified four main characteristics of a blockchain that have a direct impact on a process monitoring platform, namely: accessibility, mining strategy, programmability, and remuneration. Indeed, based on such characteristics, a monitoring platform may face specific challenges, or some challenges could be mitigated only if the blockchain has some of these characteristics. Section 7 will clarify the relationship between characteristics, challenges and mitigation strategies.

Table 1 classifies the most commonly adopted blockchain and DLT platforms, as outlined in [79], according to the aforementioned characteristics. The **accessibility** of the platform in use determines which nodes can access the distributed ledger: *private* blockchains and DLTs are accessible only to a restricted number of peers, as opposed to the *public* ones. The **mining strategy** refers to the participants who can decide on the next published transactions: if a selected number of participants is granted that right, the platform is *permissioned*, otherwise it is *permissionless*. Natively, Bitcoin, Ethereum and IOTA[4] [62] are public and permissionless, although Ethereum private networks can be created that operate within consortia, allowing only a subset of nodes to mine blocks. Hyperledger Fabric[5] [6], instead, is conceived as a consortium (private) permissioned blockchain. R3 Corda[6] [54] is a public permissioned blockchain, since it allows anybody to take part in it, but it also requires nodes to be granted access to process transactions.

Second-generation blockchains such as Ethereum and Hyperledger Fabric support so-called *smart contracts* [70], that is, executable code expressing how

[4] https://www.iota.org/. Accessed on April 13, 2022.
[5] https://www.hyperledger.org/projects/fabric. Accessed on April 13, 2022.
[6] https://www.r3.com/corda-enterprise/. Accessed on April 13, 2022.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

Table 1 Characteristics of blockchain platforms

		Blockchain/DLT platform				
Characteristic	**Level**	**Bitcoin**	**Ethereum**	**Hyperledger Fabric**	**R3 Corda**	**IOTA**
Accessibility	**Public**	×	×		×	×
	Private			×		
Mining strategy	**Permissionless**	×	×			×
	Permissioned			×	×	
Programmability	**Limited**	×				×
	Advanced		×	×	×	
Remuneration	**None**			×		
	Cooperative					×
	(Crypto-)currency	×	×		×	

business is to be conducted among contracting parties (e.g., the conditions to fulfill before digital assets can be transferred). Thereby, they offer *advanced* **programmability** capabilities. Other platforms such as Bitcoin come bundled with script languages offering *limited* expressive power. Smart contracts often require data from the world outside the blockchain sphere (e.g., financial data, weather-related information, random numbers, sensors from hardware devices). However, they cannot directly invoke external APIs. Therefore, smart contracts need software adaptors that play that interfacing role. Those artifacts are named *oracles* [55,79]. Oracles can be further classified as *software* or *hardware* oracles. Software oracles aim to extract information from programmed applications (e.g., web services), whereas hardware oracles extract data from the physical world – with the aid, e.g., of Internet of Things (IoT) devices.

To make their deployment sustainable and the maintenance of the infrastructure profitable to the involved nodes, blockchain platforms often include a **remuneration** scheme. Most of them require a fee to be paid whenever a new transaction is initiated by one of their participants. The fee can consist either of *cryptocurrency* (e.g., ETH, BTC) or *fiat currency* (e.g., US Dollar, Euro). The fee typically increases according to the amount of data being written and, for blockchains supporting smart contracts, to the complexity of the code being executed. IOTA and other blockchain platforms, instead, rely on a *cooperative* scheme rather than on fee-based mechanism. In this case, each participant must validate at least one transaction before being able to issue a new one. Finally, some solutions *do not apply any fee* – neither direct nor indirect – when processing transactions. That approach is typically adopted by private blockchain platforms.

3 Running Example

To better explain the need for process monitoring, what can be monitored with current techniques, and what could be achieved with a blockchain-based monitoring platform, we use an example taken from the logistics domain. Figure 1 shows the Business Process Model and Notation (BPMN) diagram of a shipment process.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

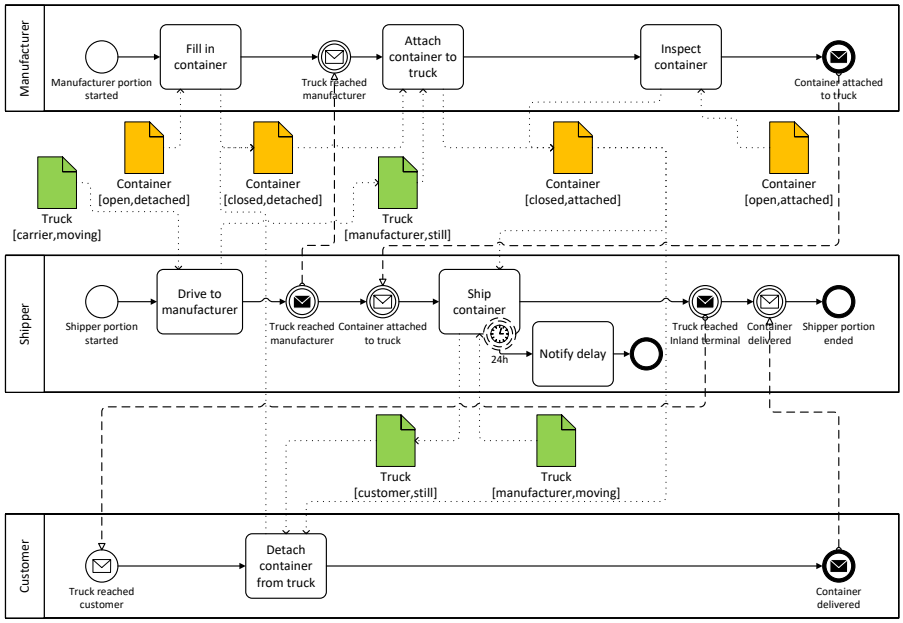

Fig. 1 Motivating example.

The example involves a manufacturer M, who receives an order from one of its customers C, and a shipper S, on whom M relies for the delivery of the goods to C. At first, M loads a shipping container with the goods requested by C. Meanwhile, S drives one of its trucks to M's production facility. Once the truck arrives, M firstly attaches the container to the truck, then inspects the container to verify that all the goods requested by C are present. The inspection should be performed only at this stage, and the container should not be opened again until it reaches the premises of C. Once the inspection completes, S ships the container to C, who detaches it from the truck. In case the shipping activity takes longer that 24 hours, S must justify the delay.

Given a multi-party business process where parties may not trust one another, the actors should at least rely on the monitoring infrastructure for process monitoring to properly work. A possible setting is to have a central authority, trusted by all the parties by means of some prior agreements, in charge of monitoring the choreography. In this case, the main drawback is the lack of flexibility of the monitoring infrastructure as it requires that all the involved actors have settled on a prior agreement, and linked all the needed software connectors, with the monitoring party. For instance, if the usual shipper S could not be temporarily involved to deliver the package to the customer C (e.g., due to a strike), the manufacturer M could not replace it with another shipper S' maintaining the same monitoring capabilities of the process, unless S' was already connected to the monitoring system.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

This is a typical setting in which a blockchain platform can be beneficial thanks to its ability to create a trusted distributed environment. Indeed, it solely requires the parties to be connected to the selected blockchain infrastructure and can be used to store in an immutable way the monitoring data.

4 Business process monitoring

Business process monitoring aims at identifying how well running processes are performing with respect to performance measures and objectives. Depending on the available tools and data, a business process platform can keep track of the running process instances, detect deviations with respect to their expected behaviour, or identify other anomalies during their execution. This section briefly introduces the main characteristics of business process monitoring platforms in terms of the possible goals of the monitoring (i.e., the why), the available techniques (i.e., the how), and the subject of monitoring (i.e., the what).

4.1 Why to monitor

There are several reasons why a monitoring platform should be introduced. Firstly, the process owner would like to know if the process is properly running to identify critical situations like bottlenecks, deadlocks, deviations from the planned execution. In fact, even if at design time techniques for verification, validation, and simulation can be put in place to produce a sound process, a real check is possible only at run-time, when the process is immersed in a real setting where unforeseen events could have an impact on a correct execution. For example, by monitoring the process in Section 3, M may find out that activity Fill in container takes much more time than Drive to manufacturer, forcing S to wait for it to conclude with an increase in time and costs. Therefore, M and S may decide to postpone activity Drive to manufacturer in order to avoid delays.

Moreover, there are cases in which the process owner needs to demonstrate to external auditors that the process is compliant with norms and regulations. In such situations, business process monitoring mechanisms are required. For example, if C finds out that some goods are missing once the process is over, M can prove to C that the container was inspected before the shipment occurred.

Finally, business process monitoring is relevant when several actors are participating in the same business process. In this case, the so-called choreographed process identifies the protocol of communication among the different actors. Business process monitoring techniques are then responsible for check the correct adoption of this protocol. For example, by monitoring the process, M can check if S started activity Ship container only after it was notified that the container was ready for shipment (i.e., after activity Inspect container was concluded).

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/s10270-021-00959-x

To achieve these objectives, as introduced in [63], the functions offered by a business process monitoring platform concern: (i) runtime monitoring, (ii) execution data post-processing, and (iii) runtime control. Runtime monitoring aims to passively track the current status of the process execution as well as of the choreography. The execution data post-processing performs analyses on top of data collected at run time to give insights to the process owner or to the participants in the choreography. Such insights rely on some indicators (e.g., Key Performance Indicators (KPIs) or Process Performance Indicators (PPIs)) to describe the behaviour of the already completed process instances. Finally, the run-time control performs an active approach to mitigate possible critical situations. To do so, it relies on process intelligence techniques which, among the others, could predict how the process will evolve. For the sake of clarity, this paper mainly focuses on the passive approach: i.e., how the blockchain can be useful to increase the trust on the monitoring data, regardless of the indicators to be computed or the possible actions to be enacted for solving critical situations. In particular, thank to the way transactions are stored in a blockchain, storing monitoring information in a blockchain would make it persistent and immutable. Thus, in case of disputes, it would be impossible for the organizations that caused such an issue to hide or alter information which proves their ill-behavior. Also, each transaction in a process contains the address (i.e., the identifier) of the issuer. Thus, the blockchain would keep track of the provenance of the monitoring information. If incorrect information is stored, it would be possible to identify the organization that introduced it. In addition, since transactions must be cryptographically signed by the issuer, it would be impossible for an organization to act on behalf of another one without consent or to repudiate the information it directly stores [29].

4.2 How to monitor

Event data logging is the generation of sequences of events related to a specific process instance being executed. Events can provide notifications on the activities being executed, or on the artifacts (i.e., the physical or virtual objects manipulated by the process) and the resources (i.e., the human operators or software components responsible for executing activities) participating in the process [68]. For example, events related to the process in Section 3 are the start of activity Fill in container and a change in the conditions of the Container artifact (e.g., when it is attached to the truck). Once collected, events are typically stored in so-called event logs. Since several other monitoring techniques require event data to work, this technique is often seen as a prerequisite for them.

Event data logging would benefit from a blockchain, as it would be possible to keep track of the provenance of events. Also, events would be persistently and immutably stored. Finally, as suggested in [79], the distributed nature of a blockchain can be used as communication mechanism, thus it would allow to easily share events related to inter-organizational business processes.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

Indeed, by writing events in the blockchain, each organization would ensure that they are propagated to the other participants who have access to the blockchain. Similarly, it would automatically receive events produced by the other participants.

Business Activity Monitoring (BAM) and runtime performance analysis: Also known as "monitoring" [1], BAM [22] analyzes real-time information on the activities being executed (e.g., response time and failure rate). With this technique it is possible to measure KPIs relevant for the process, thus determining how well activities are performed. For example, performing BAM with the process in Fig. 1 would allow the participants to know which is the minimum, maximum and average duration of activity Ship container. Instead, runtime performance analysis focuses on the data analysis of performance information on the processes being executed to identify bottlenecks or resource allocation problems [22]. Unlike BAM, which focuses on single activities, runtime performance analysis focuses on process runs, thus accounting for dependencies among activities. For example, performing runtime performance analysis on our running example would allow the participants to identify a delay that occurs after activity Ship container concludes and before activity Detach container from truck starts. Furthermore, it would identify if such a delay was caused because S did not immediately send the message informing that the container was ready for being detached, or because C did not promptly react to that message.

BAM and runtime performance analysis would benefit from the adoption of a blockchain. Indeed, the participants could agree on how the analysis should be performed by storing on-chain the parameters used by the algorithms that analyze event logs (e.g., the maximum acceptable duration of Ship container). In addition, a blockchain that offers advanced programmability would also automatically guarantee that the analysis is actually performed as agreed by the participants. Indeed, such blockchains would allow for the implementation of analysis algorithms as smart contracts [36]. Then, such smart contracts would be executed on the blockchain by all the participants responsible for validating the process. In case participants obtain different results, the consensus algorithm would automatically ensure that the correct result (i.e., the one obtained by the majority of the participants) is accepted.

Conformance checking consists in the techniques that compare the modeled process behavior with the one evidenced by execution data [2]. To this end, the gathered event data are replayed on the process model, so as to detect deviations from the expected behaviour. Given a process model and event data, Conformance checking produces conformance-related diagnostic information.

With conformance checking, the stakeholders can verify whether the execution is in line with the process description. In particular, the nature of the model plays an important role in defining the degrees of freedom that are left to the process executors. A collaboration diagram (e.g., the complete collaboration diagram in Fig. 1) will force the whole process to strictly adhere to the

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

specifications. A process diagram (e.g., only the portion of the process inside a specific pool) will force the process portion belonging to that stakeholder to adhere to the specifications. Finally, a choreography diagram will force only the interactions among stakeholders to adhere to the specifications, leaving the stakeholders free to alter their internal processes.

As for BAM and runtime performance analysis, conformance checking would benefit from the usage of a blockchain too [56,38]. Indeed, storing the process model in the blockchain would allow the participants to agree on the structure of the process, to be taken as input by the conformance checking algorithms. In addition, a blockchain that offers advanced programmability would allow for the encoding both of the process model and the conformance checking logic as smart contracts, thereby guaranteeing that conformance checking is actually performed as agreed by the participants. In addition, the smart contract could be used to raise and propagate alerts among all the participants in the process if a discrepancy between the modeled behavior and the actual process execution is detected.

Compliance checking encompasses the techniques aimed at verifying that constraints representing regulations, guidelines, policies and laws, are fulfilled by the process [35]. It differs from conformance checking because constraints focus on process rules, rather than on entire process runs.

Through compliance checking techniques, it is possible to define complex constraints on the process that predicate both on the structure and on non functional aspects. Instead of relying on a process model, compliance checking relies on compliance rules that describe only the elements of the process that are useful to assess the constraint. For example, a compliance rule related to our running example could check whether the container is delivered to C within two days since when M finished preparing it. Another compliance rule could check if less than 1 % of the shipments were carried out without inspecting the container. To this aim, according to [48], several compliance checking techniques and languages exist. Since constraints predicate on specific portions of the process, rather than on the process as a whole, it is much easier for stakeholders to agree on monitoring them. In fact, only activities required for the assessment of such constraints have to be disclosed, thus overcoming one of the issues of conformance checking.

As for conformance checking, compliance checking would benefit from the blockchain. Indeed, storing compliance rules in the blockchain would allow the participants to agree on the constraints that hold for the process. In addition, a blockchain that offers advanced programmability could encode compliance rules as smart contracts, thus making them executable on the blockchain itself and, consequently, guaranteeing that compliance checking is actually performed as agreed by the participants [36]. The smart contract could be the source of alerts for all the participants in the process whenever a compliance check fails.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

4.3 What to monitor

Depending on the monitoring technique and on the underlying representation of the process to monitor, different kinds of events have to be logged for the monitoring to be reliable. Conformance checking techniques typically require events notifying the start or termination of activities, or the transmission and receipt of messages among participants. BAM, runtime performance analysis and compliance checking techniques usually require more complex events, also indicating when artifacts were manipulated or who performed a task (e.g., starting an activity or modifying an artifact).

Depending on the scope of the monitoring and on the visibility of the running processes, different scenarios can happen as a combination of three main corner cases: monitoring (i) the activities performed by an organization, (ii) all the activities performed by all the organizations involved in the multi-party business process, or (iii) the interaction among the participants.

In case (i), the reference model adopted for process monitoring is a *process diagram* and only events belonging to the owner of the process are collected and analyzed. When the process consists only of either automated activities or form-based ones, obtaining events is a relatively easy task. In fact, event logs can be retrieved from the Business Process Management System (BPMS) in charge of executing the process. Furthermore, since users are required to interact with the BPMS to perform business activities, event logs contain accurate information on who performed which task, when the task was performed, and which artifacts were involved during its execution. However, when the process also involves manual activities, that is, activities that are performed by users without interacting with the BPMS (e.g., activity Ship container in the process in Section 3), collecting reliable event logs becomes challenging. In fact, users may forget to notify to the BPMS when they perform activities, they may incorrectly indicate in the notifications when the activities were performed, they may indicate that they performed an activity which was not done or which was done by another user. These issues can be partially solved with IoT-based solutions, such as artifact-driven monitoring [52] or Unicorn [10], which autonomously collect events from the artifacts being manipulated, to be then analyzed to infer which activity was executed.

In case (ii), when the events belonging to all the involved organizations have to be logged and shared among participants, the process to monitor is represented as a *collaboration diagram*. This is a challenging task both from the organizational and technical standpoint. From the organizational standpoint, the participants may be reluctant to share events on the activities being performed, as it may allow competitors to uncover their operations. In case of BAM and runtime performance analysis, sharing such events may even violate privacy regulations, such as the General Data Protection Regulation (GDPR), since information on the employees performing activities may be shared to the other organizations. From the technical standpoint, sharing events typically requires either individual information systems to be federated, or a centralized cross-organizational information system to be deployed and adopted by

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

all the participants. To partially overcome these issues, organizations can autonomously monitor their own portions, and then share aggregated monitoring data to the other participants. However, this approach reintroduces the problem of trust between organizations, moving it from the execution to the monitoring of the processes. In fact, for this approach to hold, organizations are required to trust each other, assuming that monitoring data reflect the actual behavior of the process.

Finally, if events related to the transmission and reception of messages between organizations have to be logged (iii), the process can be represented as a *choreography diagram*. From the technical standpoint, as long as the message exchanges are performed digitally (e.g., email, web service invocations), it is relatively easy to log and distribute events. In fact, it is sufficient to passively monitor the communication channels, generating events whenever communication activity is detected. On the other hand, if physical objects are exchanged, generating event logs is a more complex task. In fact, an active agent is required to observe the real world and produce an event whenever some physical object is either received or sent. Originally, this was done by relying on human operators, but it suffered from the same limitations as the ones outlined for manual activities. Therefore, IoT-based solutions to track physical objects are adopted as long as the contents of the messages are kept confidential, and only events relevant for the process being monitored are disclosed.

In case compliance rules have to be monitored, depending on the language and technique adopted, events related to messages, activities, or artifacts have to be logged. Consequently, compliance checking has the same technical limitations as all the conformance checking techniques. However, not every event related to the process has to be logged, but only the ones required for the verification of the compliance rules. Therefore, monitoring has a much lower footprint on the organizations. Also, since organizations can selectively choose which events are logged and made available to the other participants, they can agree on not to share information that discloses their know-how.

5 Monitoring with the blockchain: A reference architecture

To take into consideration all the characteristics of monitoring platforms described in Section 4, we build upon the BPMS reference architecture proposed in [63]. This architecture reflects the main functionalities adopted in the BPMS tools nowadays present in the literature and, in addition to elements specific to the process engine, the Business Intelligence and Analytics Suite (BPI&BPA) is identified as the component in charge of providing the monitoring facilities. As shown in Fig. 2, the BPI&BPA sub-system is composed of tools for *Data Ingestion*, an *Information Repository*, and tools for *Data Analysis* and *Data Management*. Generally speaking, Data Ingestion Tools are connected to the process engine to obtain information about the process execution and to store them, in predefined and agreed formats, into the Information Repository. The Data Management Tools access this repository to enrich the information

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

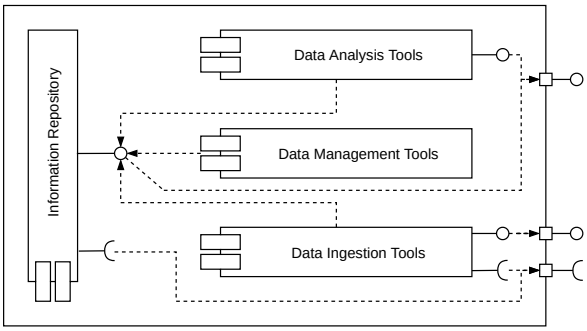

Fig. 2 An overview of the BPI&BPA component architecture [63]

(e.g., by enriching and correlating the information). Lastly, the Data Analysis Tools supports the post-processing and run-time monitoring functionalities as discussed in Section 4.

Then, to understand how the capabilities offered by a blockchain are relevant for business process monitoring, we identify the following requirements that the reference architecture of a blockchain-based monitoring platform should meet. We identify these requirements based on the characteristics that a business process monitoring system should expose, as previously discussed in Section 4.

R1. Event support: The capabilities to collect information about events is fundamental for business process monitoring (see Section 4.2, "How to monitor"). Moreover, the architecture should be independent from the monitoring techniques and from the representation of the process to monitor. Therefore, it should support events related to the execution of business processes, to the condition of the artifacts, and to the messages exchanged among the involved parties .

R2. Pluggability of participants: As the process to be monitored can involve different parties in different configurations (see the corner cases listed in Section 4.3, "What to monitor"), the architecture should be flexible enough to allow both long-term and short-term collaborations among participants. In particular, participants that establish a collaboration for the first time should easily and safely join the monitoring platform. Similarly, once a collaboration is over, participants should be able to leave the monitoring platform.

R3. Differential access to information: As illustrated by the the corner cases described in Section 4.3, the architecture should allow participants to decide which monitoring information on their own processes and artifacts should be shared among the other participants, and which information should be kept private.

R4. Access to contextual data: The architecture should allow the monitoring platform to access data that are not contained within events, but are required to interpret them. In fact, event data can be complemented

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

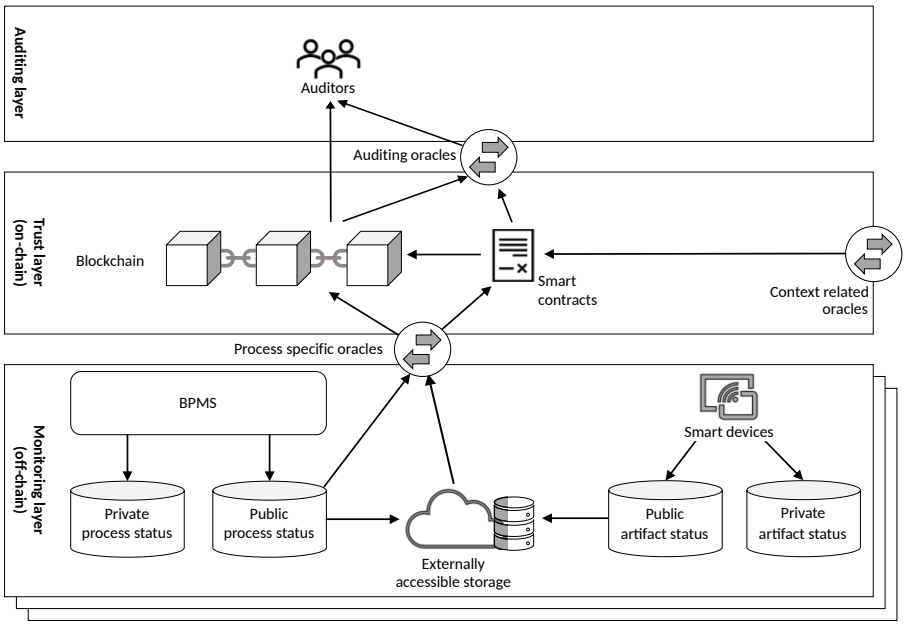

Fig. 3 Blockchain-enhanced BPMS monitoring reference architecture

with data obtained from human interventions or from IoT-based solution to provide contextual data, as discussed in Section 4.3. For example, traffic or weather data could be required by a compliance rule to determine if the shipment is taking an excessive amount of time.

R5. Auditing-oriented data provisioning: The architecture should allow for the distribution of monitoring data to external auditors, that is, subjects who do not take part in the process being monitored (see Section 4.1, "Why to monitor"). These data should be structured, enriched and aggregated to let auditors access readily usable information with limited further preprocessing needed.

To support these requirements, and taking inspiration from the reference architecture of the BPI&BPA sub-system proposed in [63], we propose the architecture shown in Fig. 3.

This architecture is organized along three main layers, i.e., *monitoring*, *trust*, and *auditing layers*, and summarizes how the different elements composing a blockchain can be combined to provide a trusted monitoring platform. In addition to the usual data collection – which is needed to know whether the process is behaving correctly or not – the blockchain provides a platform to guarantee that the monitored data cannot be counterfeited. With respect to the usual approaches, the adoption of a blockchain eliminates the need to involve third-party actors, making a blockchain-based solution more flexible.

The *monitoring layer* is fundamentally composed of both traditional and innovative tools and technologies adopted by a participant to monitor its busi-

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

ness process. The number of instances of the monitoring layer depends on the number of participants actively involved in the business process that has to be monitored. There are situations in which a single party aims to certify the information about the behaviour of its process for other actors not directly involved in the process. For example, let us suppose that the manufacturer M of our running example (Section 3) wants to verify that, for every process execution, the container is properly sealed at the end of the loading activity – so as to guarantee to the customer that nobody can access the goods while they are travelling. In a different setting, where a choreographed process is established, it is assumed that each participant offers the services included in the monitoring layer.

To address **R1**, the monitoring layer is responsible for the collection of *process status data* (i) and *artifact status data* (ii). The former concerns the control flow (e.g., when activities start or end, what events are raised during the execution, when messages are sent or received among the participants). These data are obtained by querying the event logs maintained by a process orchestrator or, in less process-aware organizations, by the systems in charge of executing the various activities. To address **R3**, the architecture distinguishes process status data obtained from event logs into *public* and *private* data. Public data refer to the status of the process, which is directly related to the agreed choreography or the process aspect that should be certified. Thus, these data have to be shared with the involved partners or the interested actors in order for them to verify if the process is running as specified in the agreed contract. Conversely, private process status data concern the execution of activities that are not seen by the other parties. The same distinction applies to artifact status data, which could rely on IoT-based applications to continuously sense the status of the goods managed during a process execution [34]. In some cases, the information collected by the paired smart devices is internally managed by the company that owns the artifact and, consequently, controls the smart devices. Conversely, when considering artifacts exchanged among the participants in a choreography, information about the status should be shared by the involved participants. For instance, let us consider the logistics scenario of Section 3 again and add that the cold-chain must be preserved. In that setting, when a package moves from the manufacturer M to the courier S, information about the temperature of the package must be shared for an effective monitoring. To conclude the features of the monitoring layer, we remark that it might happen that the public data, both about the process and the artifact, are stored on a shared space (e.g., cloud storage) easily accessible by the participants who have interest in them.[7].

The *trust layer*, where the blockchain has the major involvement, includes the set of functionalities that are used to guarantee the reliability of the data collected by the participants about the process and artifact status. Indeed, these data are intended to be used to check and demonstrate whether the

[7] For the sake of simplicity, here we assume that a proper access control system is in place to make the data visible only to the parties interested in the specific portion of the process.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

process is running as planned or not, thus to increase the trust on the process owner. In traditional approaches, trust is ensured by third parties, which are in charge of collecting – on behalf of the participant, or supervising the collection performed by the participant – data on the process and artifact status [52]. With the blockchain, this actor is no longer required, as the infrastructure provides by definition a trusted data management or, as defined in [71], the blockchain can create a trust environment even if the participants do not trust each other, thereby addressing **R2**. In particular, smart contracts can be defined based on the agreements among the participants to regulate the way in which the collected data are stored in the blockchain. Thus, they guarantee that the data are collected at the right time, in the right way, and that once stored in a block they cannot be modified. As these smart contracts operate on off-chain data (i.e., the data coming from the monitoring layer), oracles are required to enable the communication with the on-chain environment. It is worth noting that two classes of oracles are considered. On the one hand, the process specific oracles concern the elements to obtain information about the status of the process and the artifact from all the participants involved to address **R1** (e.g., considering the process in Section 3, the position of the truck, and when the container has been opened and closed). On the other hand, context related oracles are included to obtain information on the context where the process is being executed, thus addressing **R4**.

Finally, the *auditing layer* provides the perspective of whom is interested on how the process is running. Thus, auditors are the participants in the process itself which are interested in how the other participants are behaving. To this aim, they rely on the blockchain as a trusted source of information. In some cases, auditors can be the final customers, who are not involved in any of the activities, but are interested in how the goods they have bought (i.e., an artifact) are managed by the participants. In some other cases, depending on the nature of the process, auditors can be institutions (e.g., the environmental protection agency) who must be kept informed about the process being executed. When auditors are connected via a blockchain client, the blockchain supports by default a request-to-response interaction. To address **R5**, in case different messaging patterns are required, proper oracles can be established on purpose following the guidelines suggested in [56]. In this way, auditors can be informed about specific events or status changes.

Compared to the architecture of the BPI&BPA sub-system in [63], it is relevant to see how the elements of the proposed blockchain-based monitoring reference architecture fits with that (see Figure 4). Notably, the elements of the monitoring layer, along with the process specific and context related oracles, cover the functionalities required by the *Data Ingestion tools*. The *Data Management tools* corresponds to the smart contracts which drive the storage of the relevant data into the blockchain which itself, corresponds to the *Information Repository*. Finally, for the *Data Analysis Tools*, the proposed architecture offers only the auditing oracles as a way to access the data stored in the blockchain to perform any type of analysis relevant for the auditors.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

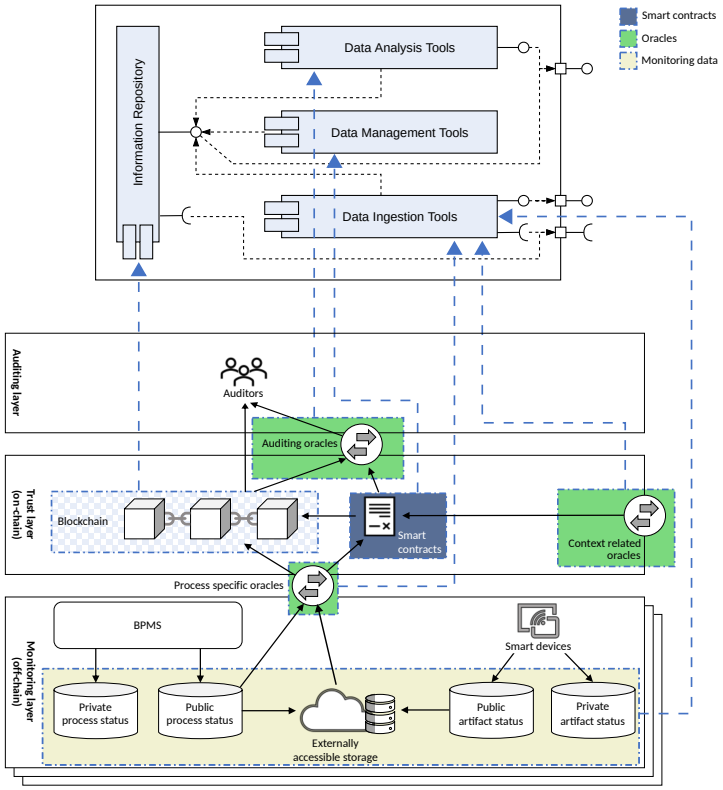

Fig. 4 Connection between BPI&BPA architecture and proposed reference architecture

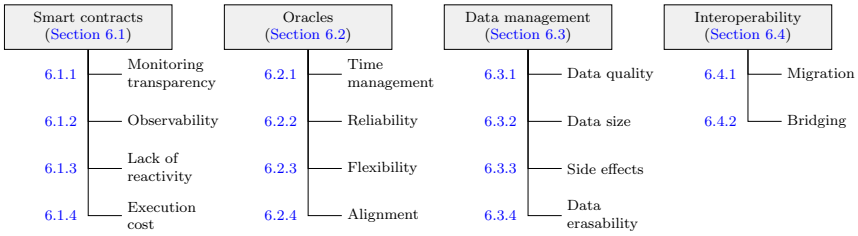

Fig. 5 Challenges for blockchain monitoring.

6 Monitoring with the blockchain: Challenges

In light of the proposed reference architecture, we identify three main component types whose design, realization and tuning can have an impact on the effectiveness and efficiency of the adoption of blockchain platforms to support business process monitoring: smart contracts, oracles, and monitoring data. In Fig. 4, they are enclosed in dark-blue, green and pale-yellow boxes, respectively. *Smart contracts*, in particular, reflect the monitoring capabilities.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

The expressiveness of the language used to define them affects the richness of these capabilities. *Oracles* identify the data sources that can be considered as trusted, since they are meant to be produced or managed on the blockchain platform. *Monitoring data* relate to the core information to be analysed – we particularly focus on the balance between the data to be stored on-chain and the data to be archived off-chain. In addition to the need for a proper design and implementation of these components, particular attention must be paid to the analysis of strategies that guarantee *interoperability* among blockchains. This is relevant especially in the case of multi-party business processes, wherein involved actors could already adopt their own blockchain solutions, thereby making the replacement (or addition) of platforms potentially counter-productive.

For each of these perspectives, we elicit a series of aspects that call for particular attention to be paid when using blockchains for business process monitoring, based on existing work on blockchain and our own experience. Figure 5 depicts the aspects that we analyze from the aforementioned perspectives. The remainder of this section discusses them in detail.

6.1 Challenges about smart contracts

6.1.1 Monitoring transparency

To improve the transparency of compliance checking, especially in the case of multi-party business processes, smart contracts hold a crucial role. In fact, based on the information that can be made accessible through oracles and on the relevant transactions mined in the blockchain, a smart contract can analyze the current status of the process enactment and verify if the control flow (in the case of orchestrated processes) or the message exchange (in the case of a choreographed process) behaves as expected. As the code composing the smart contract is executed on all the blockchain clients to reach consensus, it is extremely hard for a single party to alter it in order to counterfeit the result. As a consequence, the logic that drives compliance becomes publicly available to all the parties interested in the soundness of the process, even if they are not directly involved in the enactment (e.g., auditors). Especially in multi-party business processes, information about the obligations involving the parties can be produced and observed with smart contracts. For example, given the process in Section 3, if each of the three sub-processes modeled in the pools in Fig. 1 is implemented as a smart contract, M, S and C agree on how the process should be carried out. As the constraints become public, none of them can complain that they expected the process to be executed differently. Moreover, if the parties agree that the visibility should be limited only to the protocol governing the message exchange, then the smart contract will reflect only the choreography.

The transparency offered by the smart contracts heavily impacts on the monitoring platform. On the one hand, it increases the trust in process execu-

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

tion. On the other hand, it requires careful implementation of the *monitoring* and *trust layers* in order not to undermine the control that participants have on the data being provided to the platform, which is required to address **R3**. In particular, it requires smart contracts to be properly designed in order to rely only on the information that could be made available to external actors [12]. Moreover, the ability for a smart contract to verify possible deviations in the process enactment is strictly related to the monitoring data that are available through the oracles. As a consequence, the availability of proper data sources that can be accessed by the smart contract is fundamental. With single-party business processes, this issue can be less critical, as the party is responsible for designing the smart contracts and the oracles, as well as for choosing the necessary data sources. Multi-party business processes, instead, may require an agreement that includes the possibility to make some of the data about the process and artifact status available to the other involved parties. This opens an additional issue about the accessibility of those data. For instance, to determine if the process portion carried out by S is correctly performed, S should expose the information on the position and speed of its trucks.

6.1.2 Observability

Smart contracts and the invocations of their methods are stored in the blockchain. Their execution can thus be performed and analyzed by any participant. Additionally, most blockchains require smart contracts to explicitly define methods to retrieve their state information. In other words, variables that are used by smart contracts may be made accessible only by the smart contract itself, unless methods to make their contents available are explicitly defined in the smart contracts [42,4].

Observability issues in smart contracts could arise when implementing the *auditing layer* to distribute monitoring data to the process auditors, in order to address **R5**. Indeed, auditors may be unable to access valuable monitoring information, if mechanisms to access this information were not explicitly designed when the smart contracts were put in place. As a consequence, before putting in place a blockchain-based monitoring platform, care should be taken defining which information can be retrieved from the smart contract.

For example, suppose that, to monitor the process in Section 3, a smart contract is implemented that has an internal representation of the process and of the status of each activity. That smart contract may expose a function to check whether the process conforms to the model or not, without providing information on the activities. As a consequence, although the smart contract internally knew that, e.g., Ship container is running and Attach container to truck is complete, it would lack a way to communicate this information to other smart contracts or other participants, who cannot rely on it to determine the status of the process and its activities.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

6.1.3 Lack of reactivity

A smart contract lacks the capability of independently making calls or invocations to endpoints outside the blockchain upon the verification of certain conditions [15,64].

When implementing the *auditing layer* to keep the process stakeholders informed on the process execution (in order to address **R5**), this limitation may significantly impact the monitoring platform. Indeed, especially in case of deviations, it would be desirable that the process owner and the involved parties could promptly react to such a deviation. However, due to the lack of reactivity in smart contracts, it is not possible for a smart contract to automatically and autonomously trigger any off-chain counteraction to process deviations. For example, suppose that the smart contract monitoring the process in Section 3 detects that activity Inspect container was performed while the container was being shipped. In light of the above, it cannot autonomously contact S off-chain to request a justification for that action.

6.1.4 Execution cost

The execution of a smart contract is generally more computationally intensive than the execution of code deployed as a centralized, stand-alone application [3]. The higher expensiveness is due to several reasons. First, the code that the smart contract consists of must be executed by multiple participants in order for them to reach consensus. Second, a new block containing one or more smart contract invocations must be created for its effects to be persisted on the blockchain. Finally, the changes in the state of the blockchain caused by the invocation of a smart contract must be propagated and applied by all participants. As a consequence, the inner complexity of a smart contract is amplified by the way in which it is meant to be executed. In particular, loops and function invocations within a smart contract can significantly increase its complexity, which translates to higher execution costs in terms of hardware utilization and, consequently, actual money. In addition, to avoid a single invocation to consume all the available resources and lock-up the whole blockchain, the number of instructions executed within a single smart contract invocation is typically bound to a threshold value (e.g, the gas limit in Ethereum). If that threshold is exceeded, the corresponding invocation is terminated and any change made by that invocation on the blockchain is discarded.

When implementing the *trust layer* to analyze the events that occur during process execution (in order to address **R1**) and to integrate them with contextual information (in order to address **R4**), the potentially high and bounded cost of smart contract execution may force participants to either simplify the monitoring logic or offload it to external oracles. Therefore, care should be taken when designing smart contracts in order to balance their complexity and the guarantees that on-chain code execution provides.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

6.2 Challenges about oracles

6.2.1 Time management

Among the several aspects that are interesting to monitor about a business process, one of the most pivotal ones is checking if an activity is performed on time. Unfortunately, a blockchain lack the notion of absolute time, with the partial exception of the coarse-grained block time [56,73,41]. As a consequence, smart contracts are not readily suitable for checking that an execution occurs within a given timestamp. Although a blockchain sorts the transactions, it cannot deal with timers. This is due to the fact that the expiration of a timer, or more simply a clock-ticking event, would be an action that originates from the smart contract itself. However, as a smart contract can only perform actions that are externally invoked, such actions cannot be performed without the help of an external entity.

This limited support for time management becomes an issue for the *trust layer*. Indeed, to promptly detect issues during process execution, the monitoring smart contract may require accurate time information among the contextual information needed to address **R4**. For instance, considering the process in Section 3, a smart contract should not be employed to notify a delay if activity *Ship container* is not completed within 24 hours from its start. That smart contract cannot determine that activity *Ship container* took longer than a day until it receives a notification that the activity was completed, unless it is actively polled by an external entity.

For this reason, time has to be managed externally to the blockchain by means of specific oracles, which must be configured by the smart contract to send a trigger whenever a timeout expires. It is also important to consider that those oracles are external to the blockchain by definition, hence outside the chain of trust managed by the blockchain. Therefore, when designing a time oracle, the situation in which the oracle experiences a failure or produces fake data (e.g., it goes out of sync) has to be taken into due account.

6.2.2 Reliability

The goal of an oracle is to allow the smart contracts to acquire information from the real world. Thus, oracles are required to guarantee the correctness of the data they emit. However, this may not be the case for two reasons. First, the oracle may deliver data that are wrong or even forged (e.g., by a man-in-the-middle attack) before being sent to the smart contract. Second, the oracle may not be reliable and the data produced could be accidentally inexact [44]. Both circumstances hamper the trust in the gathered data.

Reliability issues in oracles become particularly relevant for the *trust layer*, as it requires to trust events related to the process (in order to address **R1**) and contextual information (in order to address **R4**). Indeed, incorrect data provided by an unreliable oracle may lead to misleading monitoring results. For instance, considering the process in Section 3, if the truck's GPS receiver

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

is breached or broken, the related oracle could send incorrect information on its location.

Although this issue could be partially addressed by relying on multiple oracles [9] to obtain information about a phenomenon occurring in the real world, it may not always be feasible or affordable, especially when monitoring human-based activities.

6.2.3 Flexibility

Adopting oracles to allow smart contracts to check the behaviour of a process implies that all the phenomena relevant for monitoring should be exposed through oracles. The smart contract should know in advance the addresses of the oracles providing the needed data then. However, this could result in a lack of flexibility. Adding new oracles after the monitoring has been designed could turn out to be worthless, if there is no possibility to inform the smart contract about their existence [13].

When implementing the *trust layer*, the impossibility to change at runtime the oracles providing off-chain data could significantly impair the flexibility of the monitoring platform. Indeed, supporting changes in the involved stakeholders would be required to address **R2**, and the inability to change the sources providing event data and contextual information could impair the fulfillment of, respectively, **R1** and **R4**

For example, suppose that the monitoring platform relied initially on manual notifications to determine when the container was filled in, and references to that oracle were hard-coded in the smart contracts. If later on containers are equipped with scales to automatically infer if they are full or empty, it is not possible for the platform to rely on that information, unless smart contracts are redesigned and deployed anew. Mechanisms for enabling late binding of oracles to smart contracts are thus desirable for a proper design. Notice that, without that mechanism in place, an oracle that is no longer available could not be replaced.

6.2.4 Alignment

As the blockchain alone has no visibility on the events that occur off-chain, and it is the oracle's responsibility to notify such events, a misalignment between the off-chain world state and its on-chain representation may occur.

In particular, such a misalignment may affect the oracles resposible for reporting events related to the process execution (in order to address **R1**) and the ones that provide contextual information (in order to address **R4**) to the *trust layer*. Consequently, as stated in [40], such a misalignment may lead to misleading monitoring results, especially when multiple events have to be combined in order to detect changes in the execution of the process.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

6.3 Challenges about monitoring data management

6.3.1 Data quality

Information sources queried by the oracles could be inaccurate or untrustworthy. In particular, if the data sources being used in the *monitoring layer* to provide events related to the execution of the process (in order to address **R1**), or the data sources that provide contextual information to the *trust layer* (in order to address **R4**) are poor quality ones, the monitoring platform may provide misleading results. For instance, in the case of a manual activity, e.g., the oracle is not connected to any automatic sensor thus the notification of status change for the activity is manually inserted by the operator. Consequently, the operator could mislead the system by declaring that an activity is concluded even though it is not yet the case, or by delaying the notification as an end-of-the-day operation, in which all the notifications are batched together [18].

We remark that this is a well-known problem in business process monitoring, and even the utilization of a blockchain may not be beneficial to solve it. In fact, if erroneous data are stored in the blockchain, they can be amended only by appending the correct information, as the blockchain does not allow for the alteration of data in a mined block.

6.3.2 Data size

In a blockchain, the larger the amount of stored data is, the more expensive the transaction gets. This simple rule has a significant impact on monitoring costs. Indeed, initial approaches for process data extraction [56,38] assume that the necessary information is entirely stored on-chain.

To reduce the costs, though, care should be taken in the design of the smart contracts to minimize the amount of on-chain information to the sole data that are required to enable monitoring and provide a handle to external sources for deep-diving into richer information. As a consequence, this limitation impacts on the way events related to the process execution and contextual information should be handled by the *trust layer* (and, consequently, could hinder the fulfillment of **R1** and **R4**).

6.3.3 Side effects

Most blockchain platforms are prone to soft forks, i.e., branches in the chain of blocks caused by two or more blocks pointing to the same predecessor [11]. To solve ambiguities, blockchain clients such as those of Bitcoin and Ethereum consider the longest chain as the valid one (that is, they retain the one having the highest number of subsequent blocks originating from the point of forking).

From a monitoring standpoint, this lack of information consistency is an issue, since valid monitoring data may not be considered as the block containing them happen to lie on a discarded post-fork branch. As a consequence,

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

soft forks may have a negative impact on the *trust layer*. In particular, they may cause the monitoring platform to have a partial view on the events related to the process and on the contextual information, which could hinder the fulfillment of, respectively, **R1** and **R4**.

Aside from soft forks, public blockchains such as Ethereum are also prone to so-called hard forks [50]. In case a change in the consensus protocol is made – for either technical or political reasons – some participants may not accept it. Unlike soft forks, hard forks cause a split in the blockchain network, which hampers interoperability. From the monitoring standpoint, hard forks may have a negative impact on the *trust layer*. In particular, they may break the platform if some participants decide not to migrate to the new protocol. Also, they may prevent participants that use different branches of the fork from joining the platform, thus affecting the pluggability of participants, which is required for **R2**.

6.3.4 Data erasability

By design, the blockchain impedes nodes to erase or alter data already stored in it [67]. This infrastructural characteristic allows for tamper-proof monitoring solutions, which is particularly valuable when disputes between participants arise, or when the processes must be assessed by external auditors.

However, the inability for participants to decide when monitoring information should no longer be accessible by the platform may become an issue that affects the *trust layer* by hindering the fulfillment of **R3**. Indeed, some monitoring applications require old data to be purged once their retention period is over. For example, the GDPR regulations explicitly states that any sensitive information belonging to a person should be permanently destroyed on request by that person, once the trade obligations are over [37]. In these scenarios, the inability to erase information would be a serious issue. Therefore, a trade-off between the immutability of monitoring information and the so-called right-to-be-forgotten is necessary.

6.4 Challenges about interoperability among blockchains

6.4.1 Migration

As the blockchain ecosystem is in continuous evolution, a considerable number of blockchain and DLT platforms has been proposed so far. Some of them have gained little success, and thus they have been abandoned by the community of developers and users alike.

Therefore, it may happen in the future that the blockchain adopted for the *trust layer* of the monitoring platform will eventually no longer be supported. This will have a negative impact on the *trust layer*, as the lack of support may prevent other participants from joining the platform, thus hindering the fulfillment of **R2**. Thus, to avoid a monitoring solution to be locked-in to a

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

Table 2 Currently available strategies to address the challenges in blockchain-based process monitoring.

Aspect	Challenge	Applicable blockchain				Available strategy
		Programmability	Accessibility	Mining strategy	Remuneration	
Smart contracts	Monitoring transparency	Advanced	Private	Any	Any	Access control mechanisms [33]
		Advanced	Any	Any	Any	Encryption [39]
	Observability	Advanced	Public	Any	Any	Transaction mining [21]
	Lack of reactivity	Advanced	Any	Any	Any	Outbound oracles [55]
	Execution cost	Advanced	Any	Any	Any	Certified off-chain offloading [24]
Oracles	Time management	Any	Any	Any	Any	Certification schemes [59]
	Reliability	Any	Any	Any	Any	Certification schemes [59]
	Flexibility	Advanced	Any	Any	None or Cooperation	Support for late binding [45]
	Alignment	Any	Any	Any	Any	Invocation patterns [40]
Data management	Data quality	Advanced	Any	Any	None or Cooperation	Data quality smart contracts [14]
	Data size	Any	Any	Any	Any	On-chain digest [53]
	Side effects	Any	Public	Permissionless	Any	–
	Data erasability	Any	Any	Any	Any	On-chain digest [53]
Interoperability	Migration	Any	Any	Any	Any	Blockchain gateways [25,31]
	Bridging	Any	Any	Any	Any	Blockchain relays [27]

platform that is no longer used, a strategy to easily migrate the solution over a different blockchain should be defined when designing that solution [7].

Defining such a migration strategy is not trivial, as each blockchain may implement its own consensus algorithm, block structure, data representation, and smart-contract languages at times. Therefore, a transition to a different platform can require a refactoring of several components in the original architecture, potentially involving not only software connectors and gateways but also the business logic itself.

6.4.2 Bridging

A blockchain considers as trusted only those data and smart contracts that are internal to its own environment. It is then unable to directly invoke smart contracts or access data that reside on a different deployment [66].

This limitation may have a negative impact on the *trust layer*. Often, the execution of a process can trigger or influence the execution of other processes. Therefore, platforms that are responsible for the monitoring of interrelated processes should be able to communicate with one another. However, when monitoring information is stored in two different blockchain platforms, they cannot directly interact with each other, which is required to address **R2**.

7 Mitigation and applicability of challenges

The challenges identified and discussed in the previous section can have higher or lower impact on the process monitoring solutions depending on the characteristics of the adopted blockchain. For instance, the lack of reactivity challenge affects only the blockchains with advanced programmability (e.g., Ethereum, HyperLedger Fabric and R3 Corda), while the observability affects a subset of them with accessibility as public (e.g., Ethereum and Corda). Yet, time management and reliability are relevant challenges when providing a blockchain-based business process monitoring platform, regardless of the specific adopted technology.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

In this section, we link the aspects used to classify the blockchain platforms as in Table 1 with the challenges identified in Fig. 5 and illustrate the strategies currently available in the literature to mitigate the effects, if any. Table 2 summarizes the results of our investigation. Being it at its early stages, the adoption of blockchain for supporting the monitoring activities in business process management does not provide a significant number of work available in the literature. Nevertheless, we hope that the proposed reference model and the identified challenges could provide a useful roadmap for researchers working in this topic.

Access control mechanisms. To address the challenge of controlling access to the blockchain and its assets, several solutions have been proposed [33]. However, most of them focus on on-chain access control mechanisms, as their main purpose is to prevent unauthorized users from storing data or invoking smart contracts. Unfortunately, they are scarcely effective for monitoring transparency (Section 6.1.1), as anyone who has access to the global state of the blockchain can reconstruct the on-chain data. Conversely, off-chain access control mechanisms, such as the ones that are typically implemented in private blockchains, can effectively prevent unauthorized users from accessing on-chain information. However, the granularity of the on-chain information that can be protected is often either too coarse (e.g., a subset of the nodes) or too fine (e.g., a single transaction).

Encryption. Another alternative to address the challenge of monitoring transparency (Section 6.1.1) is represented by encryption [39]. By storing on-chain data in an encrypted form, and providing decryption keys only to the users who are authorized to access monitoring data, it is possible to protect them also when they are stored in a public blockchain. In addition, key exchanges can happen while the monitoring platform is running, as soon as a specific participant has to access monitoring data. However, this approach suffers from several limitations. First, encryption is limited to data, so the logic that governs the smart contracts remains publicly visible. Second, encrypted data can be stored and retrieved by smart contracts. However, for a smart contract to process them, they must be unencrypted first, either by passing the decryption key to the smart contract, or by decrypting them off-chain and then passing the unencrypted data to the smart contract. In both cases, anyone who has access to the blockchain can access monitoring data, as all the information required to decrypt them would be available on-chain. Finally, as on-chain data are permanently stored, in case the decryption key or the encryption algorithm get compromised, a new encryption scheme cannot be applied to already stored monitoring data, nor data can be deleted or be made inaccessible. Thus, anybody could be able to decrypt and access all monitoring information that were stored before the encryption was changed.

Transaction mining. To address the challenge of smart contract observability (Section 6.1.2), process mining techniques can be applied to trace the execu-

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

tion of each transaction since when the deployment took place [21]. In this way, they can identify how the state of the smart contracts (i.e., the value of their internal variables) changes over time. The downside of this approach is that, even though it processes on-chain data, it has to be performed off-chain. Consequently, if the discovered information is required by a smart contract, this information must be provided off-chain, with consequent trust issues and the need to rely on an oracle. Also, this approach is only applicable for public blockchains, as it requires all transactions committed to the blockchain to be publicly announced.

Outbound oracles. To address the challenge of the lack of reactivity in smart contracts (Section 6.1.3), oracles conforming to the outbound pattern can be deployed [55]. Indeed, this pattern has been explicitly designed to provide on-chain information to the off-chain world. To do so, smart contracts should be designed to either expose public methods that can be periodically called by an oracle to retrieve information, or emit events and require the oracle to constantly monitor the blockchain in order to catch them as soon as they fire.

Certified off-chain offloading. To address execution cost (Section 6.1.4), a strategy could include the relocation of the most computationally expensive operations off-chain. In this way, the smart contract would solely trigger an off-chain computation and receive the results. To guarantee that the off-chain computation is performed correctly, mechanisms such as zero-knowledge proof [24] can be adopted. In particular, zero-knowledge proofs allow for a quick verification of the correctness of the results without the need to redo the computation. The downside of this approach is that its applicability and effectiveness depend on how the original computation is conducted. Alternatively, a mitigation strategy can resort on an oracle that performs the off-chain computation. This, however, would bring the challenges related to oracles (Section 6.2).

Certification schemes for oracles. To address the correct time management (Section 6.2.1) and the reliability of oracles (Section 6.2.2), a possible solution could be provided by the introduction of certification schemes for oracles. One of such schemes is the usage of a Public Key Infrastructure (PKI) to authenticate oracles and prove that they are reliable. However, such a scheme would require a manual, periodical verification. Consequently, it would not enable a prompt reaction in case an oracle gets compromised. Another alternative would be the usage of zero-knowledge proof to constantly verify the behavior of oracles. However, this scheme may suffer from the same issues we discussed for certified off-chain offloading. Also, it would require a dedicated protocol that publicly notifies the subscribers when an oracle gets compromised. A third option would be to rely on decentralized oracles [59], where several nodes composing the oracle perform the same operation, and common agreement is reached with voting mechanisms. With such a design, the effort to cheat on the smart contract would become significant as it would require

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

to forge several independent nodes. Moreover, the voting mechanism can determine which nodes cannot be trusted – under the assumption that only a minority of them incurs into problems.

Support for late binding of oracles. To increase the flexibility in oracles' invocations (Section 6.2.3), a possible solution is to decouple the oracle invocation logic from the off-chain data retrieval one. In this way, it would be possible to create a late binding between the on-chain logic and the oracle, similarly to what has already been proposed for resources [45]. Thereby, an oracle would become dynamically interchangeable while the monitoring platform is running. To that end, a possible approach would include two smart contracts: one for the invocation of the oracle (henceforth, router contract), and one for the actual data processing (henceforth, processor). In particular, the processor contains a reference to the router, which is invoked whenever off-chain data are required. Changing the oracle would require to change the reference to the router in the processor. The reference change would be performed at runtime without causing any service interruption, provided that the processor has been properly designed. The downside of this approach is that it is applicable only for blockchain platforms with advanced programmability. In addition, the complexity introduced by late binding mechanisms may have an impact on the cost of the platform, especially when remuneration is based on cryptocurrency.

Invocation patterns for oracles. To mitigate the issues related to the alignment between the off-chain world and its on-chain representation (Section 6.2.4), one can implement diverse strategies to interact with the blockchain in the oracles [40]. In this way, depending on the requirements of the monitoring application, it is possible to choose the strategy that provides the best trade-off between the need for timely off-chain data and the overhead (and consequent cost) to obtain them.

Data quality smart contracts. To mitigate data quality issues in the blockchain (Section 6.3.1), a possible solution is the implementation of data quality assessment mechanisms in smart contracts [14]. To increase flexibility, data quality mechanisms can be defined as library functions or as external smart contracts. Such libraries and contracts would be referenced or invoked, respectively, from within the smart contracts that are part of the monitoring platform. In this way, low-quality data can be identified and dealt with as soon as they are introduced in the platform. The limitations of this approach are its applicability, which is limited to blockchain platforms with advanced programmability, and the increased on-chain computational requirements that data quality controls require, which are particularly relevant when a blockchain relying on a cryptocurrency-based remuneration scheme is adopted.

On-chain digest for off-chain data. To address the challenges related to the size of monitoring data (Section 6.3.2) and their erasability (Section 6.3.4), care should be taken in the design of the smart contract to minimize the

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

amount of on-chain information to the sole data that are required to perform monitoring [38]. To this aim, monitoring data can be left in an externally accessible storage such as IPFS[8], and only a reference to those data, together with their digest, be stored on-chain [53]. In this way, the authenticity of monitoring data is preserved. By computing their digest and comparing it against the hash code of the off-chain data, participants can determine if the monitoring data have been altered. Also, when the off-chain data have to be deleted, the digest alone cannot be used to reconstruct them, thus ensuring erasability. This approach alone can only detect if monitoring data were altered. Conversely, it cannot protect data against intentional or accidental corruption. Therefore, it should be combined with traditional techniques, such as distributed storage systems, Redundant Array of Independent Disks (RAID) or Error Correction Code (ECC). Another limitation of this approach is that it cannot rely on smart contracts to process monitoring data. Since smart contracts cannot retrieve and process off-chain data, those data must be first stored on chain, thus reintroducing all the issues related to data size and erasability.

Blockchain gateways. To address the challenges posed when migrating a blockchain-based monitoring platform (Section 6.4.1), a possible strategy is the use of blockchain gateways [25,31]. A gateway provides a blockchain-agnostic protocol to deploy and invoke smart contracts. Thereby, the monitoring application does not need to interact with the blockchain and to understand its protocols and smart contract languages, as those tasks are carried out by the gateway. The limitations of this approach are the need for participants to trust the gateway, and the impossibility for it to migrate data that have already been stored in a blockchain.

Blockchain relays. To mitigate the issue of accessing data across different blockchains (Section 6.4.2), relay schemes have been introduced [27]. In particular, the relay acts as a bridge between two blockchain deployments, and the way relays work is similar to that of oracles. When a method is to be invoked or data is to be accessed, the blockchain deployment requesting that information emits an event. That event is then intercepted by the relay, which performs the requested operation on the target deployment, and then sends the result to the source deployment via a callback mechanism. As for oracles, relays suffer from similar issues: guaranteeing their reliability and the possibility for them to be replaced at runtime is challenging.

□

It is worth observing that, to the best our knowledge, no approach has been proposed thus far to mitigate issues related to side effects caused by forks in public or permissionless blockchains (Section 6.3.3). Therefore,for a monitoring platform to be immune from the side effects caused by forks, the only solution currently applicable is to rely on a private permissioned blockchain, which is not subject to forks by design.

[8] Interplanetary File System (IPFS), https://ipfs.io. Accessed: April 13, 2022.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

8 Related work

Mendling et al. [51] outline the challenges and opportunities of applying blockchain technology to Business Process Management (BPM). García-García et al. [30] also discuss the benefits of adopting blockchains for BPM, and provide a through review of the current research work in that direction. Concerning process monitoring, both papers see the challenge of integrating on-chain events related to the execution of a process with external data sources, in order to obtain complete and reliable event logs. They outline the advantages that a blockchain-based monitoring platform would bring too, especially when performing compliance and conformance checking of distributed business processes.

Regarding the implementation of process monitoring platforms, to date, preliminary attempts have been proposed that can be the basis to be built upon for process monitoring in the blockchain. Smart contracts allow for the codification of business process logic on the blockchain, as shown in the seminal work of Weber et al. [76]. Later, a similar approach has been applied within the Caterpillar [47] and Lorikeet [74] tools, as well as by Madsen et al. [49] and by Corradini et al. [17]. As several modern BPMSs do, those approaches adopt a Model-driven Engineering (MDE) paradigm to let the process analysts provide graphical representations of the process and turn it into executable code enacting it [19]. López-Pintado et al. [46] propose a blockchain-based BPMS that relies on an interpreted process model. Instead of being transformed into a smart contract, the process model is parsed by existing smart contracts, as it would typically happen with a traditional BPMS.

From the monitoring perspective, the efforts have been mostly devoted to event data logging thus far. To this aim, two complementary strategies have been proposed in the literature: (i) collecting events originating outside the blockchain to create on-chain execution logs, and (ii) creating off-chain execution logs by analyzing events that occurred on the blockchain.

Concerning strategy (i), Meroni et al. [53] rely on the blockchain and on the IoT to collect information on the execution of multi-party processes. In particular, the smart objects in the process use the blockchain to communicate events on their own conditions with each other. Those events are then analyzed by off-chain monitoring techniques, such as artifact-driven monitoring. Similarly, Alves et al. [5] and Sturm et al. [69] extend a BPMS with a connector which stores the process state in the blockchain. Viriyasitavat et al. [75] exploit the blockchain to perform service selection by collecting Quality of Service (QoS) parameters.

As for strategy (ii), the main rationale is to extract and process the payload of transactions to turn them into event logs that are readily available for process mining tools. The ordering of events is based upon the ordering of the transactions in the ledger, whereas the attributes of the event (activity name, timestamp, resource, and the like) are identified based on the signature of the invoked function on the smart contract [56], a user-defined descriptor (manifest) [38], or the change of the smart contract's attribute value [21].

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

Thereupon, process mining techniques (including conformance checking) are held to analyse the generated event logs.

Other approaches have been applied to analyse blockchain-mediated communications among peers, such as GraphSense [32]. Filtz et al. [26] studied the graph of addresses in Bitcoin and thereby examined the transaction behavior of users, taking into consideration exchange rates between virtual and fiat currencies. Prybila et al. [65] focus in particular on the transposition of handovers of tasks in a process to Bitcoin transactions. With their software prototype, they are thus able to verify the execution flow of a process by tracking the transactions exchanged among peers.

Concerning the impact of blockchain in the design of applications, Lo et al. [43] propose a guided method to identify whether blockchains fit the application requirements. Similarly, Xu et al. [80] define a taxonomy to classify blockchains and propose a model to guide the designers in choosing the blockchain that best meets their needs. In [78,8,23], design patterns are proposed to help software architects in designing blockchain-based applications. Müller et al. [57] provide a discussion on how the blockchain can improve trust in collaborative processes, and introduce design patterns specifically devised to increase trust. To our knowledge, save from our previous work [20], we are not aware of any other publication that specifically addresses the challenges and provided mitigation strategies for blockchain-based process monitoring.

9 Discussion and threats to validity

Although the adoption of blockchain to achieve trusted BPM has been investigated by the scientific community and the industry for a few years, it has mostly focused on process execution. Conversely, few research work aiming at process monitoring exist. Therefore, this work is to our knowledge the first to propose a reference architecture for trusted process monitoring, as well as to provide a systematic analysis of the challenges that blockchain introduces. However, this work is not immune from threats to validity. Taking inspiration from the seminal classification of Zhou et al. [81], we acknowledge that the following ones apply to our work:

Restricted time span: The inability of the researcher to anticipate relevant studies outside the time span defined and prepared in the planning phase. As the blockchain is a continuously evolving technology, several new applications and technologies are introduced every day. Therefore, this article may have not taken into consideration some blockchain implementations or other studies that managed to mitigate or solve the challenges we identified, simply because they were not publicly available at the time of writing.

Bias in study selection: The subjective conjecture that researchers have when selecting research work relevant for a publication. This may result in excluding research work that may be relevant for this publication, or in including unrelated research work. In particular, this may have occurred due to the different knowledge, experience, and expectations that we have in

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

the field of blockchain. To minimize this threat to validity, each finding was analyzed by all the authors of this paper and included only when general agreement was reached.

Bias in data extraction: The subjective conjecture that researchers have when interpreting findings in the related work. Being most of the available blockchain implementations either described at a very high level for marketing purposes, or at very low level for developers to implement custom solutions, we may have missed some details or incorrectly understood them. As for bias in data extraction, this threat to validity was minimized by having all authors check all notions and concepts and include them only when all agreed on their description and categorization.

Primary study generalizability: The impossibility for the research study to be applicable to contexts other than the one where the study was taken. This may result in the findings of this publication to be inaccurate or irrelevant for a monitoring platform relying on a blockchain implementation different from the ones analyzed in this publication. To minimize this threat to validity, we applied the following strategies. First, we verified that each identified challenge affected all the most adopted blockchain implementations. Second, we verified for each challenge that no blockchain implementation specifically aiming at addressing that challenge exists.

Lack of expert evaluation: Conclusions or results should be evaluated by an expert who understands and interpret the outcome. In this study, an evaluation by experts in academia and industry would be particularly valuable to assess the introduced concepts especially from two standpoints: verification (i.e., checking whether the proposals and notions are practically applicable and theoretically of factually supported) and validation (i.e., dictating whether challenges, constraints or alternative solutions were overlooked) [28]. To decrease the foreseeable impact of this threat to validity, we have followed an approach ispired by architecture design in software engineering [60]. We have based this study on the joint examination of the scientific literature in the fields of process monitoring and blockchain technologies and applications. We elicited the requirements from the characteristics a process-monitoring system should expose, and synthesized in an architecture blueprint the abstract system that would meet them resorting to the blockchain, also guided by the expertise of the authors of this article and the precious contribution by the reviewers in the process of writing and revising this paper. Nevertheless, the involvement of more experts is key and draws our plans for future work.

10 Conclusion

In this article, we provided a thorough discussion on blockchain-based process monitoring, outlining its advantages and the research challenges it brings. In particular, we explained the need for process monitoring in distributed business processes, and we outlined the insights that a process monitoring platform

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

should provide, together with the information they require. Starting from those elements, we discussed the advantages that blockchain would bring, and we designed a reference architecture for blockchain-based process monitoring. We then focused on the research challenges that a blockchain-based monitoring platform would face, and we discussed how existing research work can mitigate some of them under certain conditions.

Future work will focus on how to address the research challenges we identified along two complementary directions. Firstly, we aim at extending the applicability of existing techniques to other blockchains. As we observed, the concept of smart contract is key in our setting and multiple blockchain platforms are evolving towards the support of highly expressive dedicated languages, including Algorand [16] and IOTA [61]. Secondly, we plan to propose entirely new techniques to mitigate or possibly solve the issues that have been currently not addressed yet.

Also, we plan to validate the reference architecture by implementing a prototype monitoring platform that realizes it. We aim to highlight how existing blockchain-based monitoring platforms can be framed inside this architecture. This investigation will make it possible to quantify the overhead of a blockchain-based monitoring platform with respect to a traditional one, both in terms of costs and performance, and conduct an informed cost-benefit analysis for the adoption of the former.

We hope that researchers and practitioners will become aware of the opportunities and challenges to be faced when designing a blockchain-based process monitoring platform, and of the currently available strategies to circumvent the hurdles or mitigate their effect. By taking inspiration from this article, they could discover and propose additional workarounds or higlight further issues that may arise from specific use cases. Similarly, we expect alternative architectures to be proposed in the future for blockchain-based process monitoring or new mitigation strategies that exploit future advancements both in blockchains and process monitoring.

Domain experts in vertical sectors (e.g., logistics, healthcare, etc.) who may want to consider the adoption of a blockchain to support the monitoring of their processes could also benefit from the reading of this article. Indeed, by becoming aware both of the advantages brought by the blockchain in process monitoring and of the challenges it presents, they could exploit these additional criteria to support their decision, which is often driven by strategical reasons and technology push.

Acknowledgements.

The work of G. Meroni and P. Plebani was partly supported by the ITS Italy 2020 cluster under grant number "CTN01_00176_166195". The work of C. Di Ciccio was supported by the MUR under grant "Dipartimenti di eccellenza 2018-2022" of the Department of Computer Science at Sapienza University of Rome and by the Sapienza research project SPECTRA.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

References

1. van der Aalst, W.M.P.: Business process management: A comprehensive survey. ISRN Software Engineering **2013**(507984), 37 (2013), http://dx.doi.org/10.1155/2013/507984
2. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016), https://doi.org/10.1007/978-3-662-49851-4
3. Adler, F., Kitzmann, D., Jansen, M.: Analysis of costs for smart contract execution. In: Prieto, J., Pinto, A., Das, A.K., Ferretti, S. (eds.) Blockchain and Applications. pp. 153–156. Springer International Publishing, Cham (2020)
4. Ahrendt, W., Bubel, R., Ellul, J., Pace, G.J., Pardo, R., Rebiscoul, V., Schneider, G.: Verification of smart contract business logic - exploiting a java source code verifier. In: Hojjat, H., Massink, M. (eds.) Fundamentals of Software Engineering - 8th International Conference, FSEN 2019, Tehran, Iran, May 1-3, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11761, pp. 228–243. Springer (2019), https://doi.org/10.1007/978-3-030-31517-7_16
5. Alves, P.H.C., Paskin, R., Frajhof, I.Z., Miranda, Y.R., Jardim, J.G., Cardoso, J.J.B., Tress, E.H.H., da Cunha, R.F., Nasser, R., Robichez, G.: Exploring blockchain technology to improve multi-party relationship in business process management systems. In: Filipe, J., Smialek, M., Brodsky, A., Hammoudi, S. (eds.) Proceedings of the 22nd International Conference on Enterprise Information Systems, ICEIS 2020, Prague, Czech Republic, May 5-7, 2020, Volume 2. pp. 817–825. SCITEPRESS (2020), https://doi.org/10.5220/0009565108170825
6. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A.D., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolic, M., Cocco, S.W., Yellick, J.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Oliveira, R., Felber, P., Hu, Y.C. (eds.) Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018. pp. 30:1–30:15. ACM (2018), https://doi.org/10.1145/3190508.3190538
7. Bandara, H.M.N.D., Xu, X., Weber, I.: Patterns for blockchain data migration. In: EuroPLoP '20: European Conference on Pattern Languages of Programs 2020, Virtual Event, Germany, 1-4 July, 2020. pp. 7:1–7:19. ACM (2020), https://doi.org/10.1145/3424771.3424796
8. Bartoletti, M., Pompianu, L.: An empirical analysis of smart contracts: Platforms, applications, and design patterns. In: Brenner, M., Rohloff, K., Bonneau, J., Miller, A., Ryan, P.Y.A., Teague, V., Bracciali, A., Sala, M., Pintore, F., Jakobsson, M. (eds.) Financial Cryptography and Data Security - FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10323, pp. 494–509. Springer (2017), https://doi.org/10.1007/978-3-319-70278-0_31
9. Basile, D., Goretti, V., Ciccio, C.D., Kirrane, S.: Enhancing blockchain-based processes with decentralized oracles. In: Enríquez, J.G., Debois, S., Fettke, P., Plebani, P., van de Weerd, I., Weber, I. (eds.) Business Process Management: Blockchain and Robotic Process Automation Forum - BPM 2021 Blockchain and RPA Forum, Rome, Italy, September 6-10, 2021, Proceedings. Lecture Notes in Business Information Processing, vol. 428, pp. 102–118. Springer (2021), https://doi.org/10.1007/978-3-030-85867-4_8
10. Beyer, J., Kuhn, P., Hewelt, M., Mandal, S., Weske, M.: Unicorn meets chimera: Integrating external events into case management. In: Proceedings of the BPM Demo Track 2016 Co-located with the 14th International Conference on Business Process Management (BPM 2016), Rio de Janeiro, Brazil, September 21, 2016. CEUR Workshop Proceedings, vol. 1789, pp. 67–72. CEUR-WS.org (2016), http://ceur-ws.org/Vol-1789
11. Biais, B., Bisière, C., Bouvard, M., Casamatta, C.: The Blockchain Folk Theorem. The Review of Financial Studies **32**(5), 1662–1715 (04 2019), https://doi.org/10.1093/rfs/hhy095
12. Bowen, H., Yi, L., Li, F., Xinhua, D., Ping, C.: Blockchain-based access control data distribution system. In: 2019 IEEE 5th International Conference on Computer and Communications (ICCC). pp. 1231–1236 (2019), https://doi.org/10.1109/ICCC47050.2019.9064149

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/s10270-021-00959-x

13. Breiki, H.A., Rehman, M.H.U., Salah, K., Svetinovic, D.: Trustworthy blockchain oracles: Review, comparison, and open research challenges. IEEE Access **8**, 85675–85685 (2020), https://doi.org/10.1109/ACCESS.2020.2992698
14. Cappiello, C., Comuzzi, M., Daniel, F., Meroni, G.: Data quality control in blockchain applications. In: BPM (Blockchain and CEE Forum). pp. 166–181 (2019)
15. Carminati, B., Rondanini, C., Ferrari, E.: Confidential business process execution on blockchain. In: 2018 IEEE International Conference on Web Services, ICWS 2018, San Francisco, CA, USA, July 2-7, 2018. pp. 58–65. IEEE (2018), https://doi.org/10.1109/ICWS.2018.00015
16. Chen, J., Micali, S.: Algorand: A secure and efficient distributed ledger. Theor. Comput. Sci. **777**, 155–183 (2019). https://doi.org/10.1016/j.tcs.2019.02.001, https://doi.org/10.1016/j.tcs.2019.02.001
17. Corradini, F., Marcelletti, A., Morichetta, A., Polini, A., Re, B., Tiezzi, F.: Engineering trustable choreography-based systems using blockchain. In: Hung, C., Cerný, T., Shin, D., Bechini, A. (eds.) SAC '20: The 35th ACM/SIGAPP Symposium on Applied Computing, online event, [Brno, Czech Republic], March 30 - April 3, 2020. pp. 1470–1479. ACM (2020), https://doi.org/10.1145/3341105.3373988
18. Denisov, V., Fahland, D., van der Aalst, W.M.P.: Unbiased, fine-grained description of processes performance from event data. In: Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11080, pp. 139–157. Springer (2018), https://doi.org/10.1007/978-3-319-98648-7_9
19. Di Ciccio, C., Cecconi, A., Dumas, M., García-Bañuelos, L., López-Pintado, O., Lu, Q., Mendling, J., Ponomarev, A., Binh Tran, A., Weber, I.: Blockchain support for collaborative business processes. Informatik Spektrum **42**, 182–190 (May 2019), https://doi.org/10.1007/s00287-019-01178-x
20. Di Ciccio, C., Meroni, G., Plebani, P.: Business process monitoring on blockchains: Potentials and challenges. In: Enterprise, Business-Process and Information Systems Modeling - 21st International Conference, BPMDS 2020, 25th International Conference, EMMSAD 2020, Held at CAiSE 2020, Grenoble, France, June 8-9, 2020, Proceedings. Lecture Notes in Business Information Processing, vol. 387, pp. 36–51. Springer (2020), https://doi.org/10.1007/978-3-030-49418-6_3
21. Duchmann, F., Koschmider, A.: Validation of smart contracts using process mining. In: ZEUS. pp. 13–16 (2019), http://ceur-ws.org/Vol-2339/paper3.pdf
22. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, Second Edition. Springer (2018), https://doi.org/10.1007/978-3-662-56509-4
23. Eberhardt, J., Tai, S.: On or off the blockchain? insights on off-chaining computation and data. In: Paoli, F.D., Schulte, S., Johnsen, E.B. (eds.) Service-Oriented and Cloud Computing - 6th IFIP WG 2.14 European Conference, ESOCC 2017, Oslo, Norway, September 27-29, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10465, pp. 3–15. Springer (2017), https://doi.org/10.1007/978-3-319-67262-5_1
24. Eberhardt, J., Tai, S.: Zokrates - scalable privacy-preserving off-chain computations. In: IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), iThings/GreenCom/CPSCom/SmartData 2018, Halifax, NS, Canada, July 30 - August 3, 2018. pp. 1084–1091. IEEE (2018), https://doi.org/10.1109/Cybermatics_2018.2018.00199
25. Falazi, G., Breitenbücher, U., Daniel, F., Lamparelli, A., Leymann, F., Yussupov, V.: Smart contract invocation protocol (SCIP): A protocol for the uniform integration of heterogeneous blockchain smart contracts. In: CAiSE. Lecture Notes in Computer Science, vol. 12127, pp. 134–149. Springer (2020)
26. Filtz, E., Polleres, A., Karl, R., Haslhofer, B.: Evolution of the bitcoin address graph. In: Data Science – Analytics and Applications. pp. 77–82. Springer (2017)
27. Frauenthaler, P., Sigwart, M., Spanring, C., Schulte, S.: Testimonium: A cost-efficient blockchain relay. CoRR **abs/2002.12837** (2020), https://arxiv.org/abs/2002.12837

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/s10270-021-00959-x

28. Frederiks, P., van der Weide, T.: Information modeling: The process and the required competencies of its participants. Data & Knowledge Engineering **58**(1), 4 – 20 (2006), http://doi.org/10.1016/j.datak.2005.05.007
29. Fridgen, G., Radszuwill, S., Urbach, N., Utz, L.: Cross-organizational workflow management using blockchain technology - towards applicability, auditability, and automation. In: Bui, T. (ed.) 51st Hawaii International Conference on System Sciences, HICSS 2018, Hilton Waikoloa Village, Hawaii, USA, January 3-6, 2018. pp. 1–10. ScholarSpace / AIS Electronic Library (AISeL) (2018), http://hdl.handle.net/10125/50332
30. García-García, J.A., Sánchez-Gómez, N., Lizcano, D., Cuaresma, M.J.E., Wojdynski, T.: Using blockchain to improve collaborative business process management: Systematic literature review. IEEE Access **8**, 142312–142336 (2020), https://doi.org/10.1109/ACCESS.2020.3013911
31. Hardjono, T., Lipton, A., Pentland, A.: Towards a design philosophy for interoperable blockchain systems. CoRR **abs/1805.05934** (2018), http://arxiv.org/abs/1805.05934
32. Haslhofer, B., Karl, R., Filtz, E.: O bitcoin where art thou? insight into large-scale transaction graphs. In: SEMANTiCS (Posters, Demos) (2016), http://ceur-ws.org/Vol-1695/paper20.pdf
33. Iqbal, M., Matulevicius, R.: Comparison of blockchain-based solutions to mitigate data tampering security risk. In: Business Process Management: Blockchain and Central and Eastern Europe Forum - BPM 2019 Blockchain and CEE Forum, Vienna, Austria, September 1-6, 2019, Proceedings. Lecture Notes in Business Information Processing, vol. 361, pp. 13–28. Springer (2019), https://doi.org/10.1007/978-3-030-30429-4_2
34. Janiesch, C., Koschmider, A., Mecella, M., Weber, B., Burattin, A., Di Ciccio, C., Gal, A., Kannengiesser, U., Mannhardt, F., Mendling, J., Oberweis, A., Reichert, M., Rinderle-Ma, S., Song, W., Su, J., Torres, V., Weidlich, M., Weske, M., Zhang, L.: The internet-of-things meets business process management: Mutual benefits and challenges. IEEE Systems, Man, and Cybernetics Magazine **6**(4), 34–44 (2020), https://doi.org/10.1109/MSMC.2020.3003135
35. Kharbili, M.E., de Medeiros, A.K.A., Stein, S., van der Aalst, W.M.P.: Business process compliance checking: Current state and future challenges. In: Loos, P., Nüttgens, M., Turowski, K., Werth, D. (eds.) Modellierung betrieblicher Informationssysteme - Modellierung zwischen SOA und Compliance Management - 27.-28. November 2008 Saarbrücken, Germany. LNI, vol. P-141, pp. 107–113. GI (2008), https://dl.gi.de/20.500.12116/23621
36. Kirrane, S., Di Ciccio, C.: Blockconfess: Towards an architecture for blockchain constraints and forensics. In: IEEE International Conference on Blockchain, Blockchain 2020, Rhodes, Greece, November 2-6, 2020. pp. 539–544. IEEE (2020), https://doi.org/10.1007/978-3-030-85867-4_8
37. Kirrane, S., Villata, S., d'Aquin, M.: Privacy, security and policies: A review of problems and solutions with semantic web technologies. Semantic Web **9**(2), 153–161 (2018), https://doi.org/10.3233/SW-180289
38. Klinkmüller, C., Ponomarev, A., Tran, A.B., Weber, I., van der Aalst, W.: Mining blockchain processes: Extracting process mining data from blockchain applications. In: BPM (Blockchain and CEE Forum). pp. 71–86 (2019), https://doi.org/10.1007/978-3-030-30429-4_6
39. Köpke, J., Franceschetti, M., Eder, J.: Balancing privity and enforceability of bpm-based smart contracts on blockchains. In: Business Process Management: Blockchain and Central and Eastern Europe Forum - BPM 2019 Blockchain and CEE Forum, Vienna, Austria, September 1-6, 2019, Proceedings. Lecture Notes in Business Information Processing, vol. 361, pp. 87–102. Springer (2019), https://doi.org/10.1007/978-3-030-30429-4_7
40. Ladleif, J., Weber, I., Weske, M.: External data monitoring using oracles in blockchain-based process execution. In: Business Process Management: Blockchain and Robotic Process Automation Forum. pp. 67–81. Springer International Publishing, Cham (2020)
41. Ladleif, J., Weske, M.: Time in blockchain-based process execution. In: 24th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2020, Eindhoven, The Netherlands, October 5-8, 2020. pp. 217–226. IEEE (2020), https://doi.org/10.1109/EDOC49727.2020.00034

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/s10270-021-00959-x

42. Liu, B., Sun, S., Szalachowski, P.: SMACS: smart contract access control service. In: 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020. pp. 221–232. IEEE (2020), https://doi.org/10.1109/DSN48063.2020.00039
43. Lo, S.K., Xu, X., Chiam, Y.K., Lu, Q.: Evaluating suitability of applying blockchain. In: 22nd International Conference on Engineering of Complex Computer Systems, ICECCS 2017, Fukuoka, Japan, November 5-8, 2017. pp. 158–161. IEEE Computer Society (2017), https://doi.org/10.1109/ICECCS.2017.26
44. Lo, S.K., Xu, X., Staples, M., Yao, L.: Reliability analysis for blockchain oracles. Comput. Electr. Eng. **83**, 106582 (2020), https://doi.org/10.1016/j.compeleceng.2020.106582
45. López-Pintado, O., Dumas, M., García-Bañuelos, L., Weber, I.: Dynamic role binding in blockchain-based collaborative business processes. In: Advanced Information Systems Engineering - 31st International Conference, CAiSE 2019, Rome, Italy, June 3-7, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11483, pp. 399–414. Springer (2019), https://doi.org/10.1007/978-3-030-21290-2_25
46. López-Pintado, O., Dumas, M., García-Bañuelos, L., Weber, I.: Interpreted execution of business process models on blockchain. In: 23rd IEEE International Enterprise Distributed Object Computing Conference, EDOC 2019, Paris, France, October 28-31, 2019. pp. 206–215. IEEE (2019), https://doi.org/10.1109/EDOC.2019.00033
47. López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I., Ponomarev, A.: Caterpillar: A business process execution engine on the ethereum blockchain. Softw., Pract. Exper. **49**(7), 1162–1193 (2019), https://doi.org/10.1002/spe.2702
48. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.P.: Compliance monitoring in business processes: Functionalities, application, and tool-support. Information Systems **54**, 209–234 (2015), https://doi.org/10.1016/j.is.2015.02.007
49. Madsen, M.F., Gaub, M., Høgnason, T., Kirkbro, M.E., Slaats, T., Debois, S.: Collaboration among adversaries: Distributed workflow execution on a blockchain. In: FAB. pp. 8–15 (2018)
50. Mehar, M.I., Shier, C.L., Giambattista, A., Gong, E., Fletcher, G., Sanayhie, R., Kim, H.M., Laskowski, M.: Understanding a revolutionary and flawed grand experiment in blockchain: The DAO attack. J. Cases Inf. Technol. **21**(1), 19–32 (2019), https://doi.org/10.4018/JCIT.2019010102
51. Mendling, J., Weber, I., Aalst, W.V.D., Vom Brocke, J., Cabanillas, C., Daniel, F., et al.: Blockchains for business process management - challenges and opportunities. ACM Trans. Manage. Inf. Syst. **9**(1), 4:1–4:16 (Feb 2018), https://doi.org/10.1145/3183367
52. Meroni, G., Baresi, L., Montali, M., Plebani, P.: Multi-party business process compliance monitoring through iot-enabled artifacts. Inf. Sys. **73**, 61 – 78 (2018), https://doi.org/10.1016/j.is.2017.12.009
53. Meroni, G., Plebani, P., Vona, F.: Trusted artifact-driven process monitoring of multi-party business processes with blockchain. In: Business Process Management: Blockchain and Central and Eastern Europe Forum - BPM 2019 Blockchain and CEE Forum, Vienna, Austria, September 1-6, 2019, Proceedings. Lecture Notes in Business Information Processing, vol. 361, pp. 55–70. Springer (2019), https://doi.org/10.1007/978-3-030-30429-4_5
54. Mohanty, D.: R3 Corda for Architects and Developers: With Case Studies in Finance, Insurance, Healthcare, Travel, Telecom, and Agriculture. Apress (2019)
55. Mühlberger, R., Bachhofner, S., Castelló Ferrer, E., Di Ciccio, C., Weber, I., Wöhrer, M., Zdun, U.: Foundational oracle patterns: Connecting blockchain to the off-chain world. In: Business Process Management: Blockchain and Robotic Process Automation Forum. pp. 35–51. Springer International Publishing, Cham (2020), https://doi.org/10.1007/978-3-030-58779-6_3
56. Mühlberger, R., Bachhofner, S., Di Ciccio, C., García-Bañuelos, L., López-Pintado, O.: Extracting event logs for process mining from data stored on the blockchain. In: BPM Workshops. pp. 690–703 (2019), https://doi.org/10.1007/978-3-030-37453-2_55
57. Müller, M., Ostern, N., Rosemann, M.: Silver bullet for all trust issues? blockchain-based trust patterns for collaborative business processes. In: Asatiani, A., García, J.M.,

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/s10270-021-00959-x

Helander, N., Jiménez-Ramírez, A., Koschmider, A., Mendling, J., Meroni, G., Reijers, H.A. (eds.) Business Process Management: Blockchain and Robotic Process Automation Forum - BPM 2020 Blockchain and RPA Forum, Seville, Spain, September 13-18, 2020, Proceedings. Lecture Notes in Business Information Processing, vol. 393, pp. 3–18. Springer (2020), https://doi.org/10.1007/978-3-030-58779-6_1
58. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), https://bitcoin.org/bitcoin.pdf, accessed: 2018-04-07
59. Nelaturu, K., Adler, J., Merlini, M., Berryhill, R., Veira, N., Poulos, Z., Veneris, A.G.: On public crowdsource-based mechanisms for a decentralized blockchain oracle. IEEE Trans. Engineering Management **67**(4), 1444–1458 (2020), https://doi.org/10.1109/TEM.2020.2993673
60. Perry, D.E., Wolf, A.L.: Foundations for the study of software architecture. SIGSOFT Softw. Eng. Notes **17**(4), 40–52 (Oct 1992), https://doi.org/10.1145/141874.141884
61. Popov, S.: IOTA: feeless and free. IEEE Blockchain Technical Briefs (2019)
62. Popov, S., Saa, O., Finardi, P.: Equilibria in the tangle. Comput. Ind. Eng. **136**, 160–172 (2019), https://doi.org/10.1016/j.cie.2019.07.025
63. Pourmirza, S., Peters, S., Dijkman, R.M., Grefen, P.: BPMS-RA: A novel reference architecture for business process management systems. ACM Trans. Internet Techn. **19**(1), 13:1–13:23 (2019), https://doi.org/10.1145/3232677
64. Praitheeshan, P., Pan, L., Doss, R.: Security evaluation of smart contract-based on-chain ethereum wallets. In: Kutylowski, M., Zhang, J., Chen, C. (eds.) Network and System Security - 14th International Conference, NSS 2020, Melbourne, VIC, Australia, November 25-27, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12570, pp. 22–41. Springer (2020), https://doi.org/10.1007/978-3-030-65745-1_2
65. Prybila, C., Schulte, S., Hochreiner, C., Weber, I.: Runtime verification for business processes utilizing the bitcoin blockchain. FGCS (2017), https://doi.org/10.1016/j.future.2017.08.024
66. Schulte, S., Sigwart, M., Frauenthaler, P., Borkowski, M.: Towards blockchain interoperability. In: Di Ciccio, C., Gabryelczyk, R., García-Bañuelos, L., Hernaus, T., Hull, R., Stemberger, M.I., Ko, A., Staples, M. (eds.) Business Process Management: Blockchain and Central and Eastern Europe Forum - BPM 2019 Blockchain and CEE Forum, Vienna, Austria, September 1-6, 2019, Proceedings. Lecture Notes in Business Information Processing, vol. 361, pp. 3–10. Springer (2019), https://doi.org/10.1007/978-3-030-30429-4_1
67. Scriber, B.A.: A framework for determining blockchain applicability. IEEE Softw. **35**(4), 70–77 (2018), https://doi.org/10.1109/MS.2018.2801552
68. Soffer, P., Hinze, A., Koschmider, A., Ziekow, H., Di Ciccio, C., Koldehofe, B., Kopp, O., Jacobsen, H., Sürmeli, J., Song, W.: From event streams to process models and back: Challenges and opportunities. Inf. Syst. **81**, 181–200 (2019), https://doi.org/10.1016/j.is.2017.11.002
69. Sturm, C., Szalanczi, J., Schönig, S., Jablonski, S.: A lean architecture for blockchain based decentralized process execution. In: Daniel, F., Sheng, Q.Z., Motahari, H. (eds.) Business Process Management Workshops - BPM 2018 International Workshops, Sydney, NSW, Australia, September 9-14, 2018, Revised Papers. Lecture Notes in Business Information Processing, vol. 342, pp. 361–373. Springer (2018), https://doi.org/10.1007/978-3-030-11641-5_29
70. Szabo, N.: Formalizing and securing relationships on public networks. First Monday **2**(9) (1997), http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/548
71. Tai, S.: Continuous, trustless, and fair: Changing priorities in services computing. In: Advances in Service-Oriented and Cloud Computing - Workshops of ESOCC 2016, Vienna, Austria, September 5-7, 2016, Revised Selected Papers. Communications in Computer and Information Science, vol. 707, pp. 205–210. Springer (2016), https://doi.org/10.1007/978-3-319-72125-5_16
72. Tasca, P., Tessone, C.J.: A taxonomy of blockchain technologies: Principles of identification and classification. Ledger **4** (2019), https://doi.org/10.5195/ledger.2019.140
73. Tikhomirov, S., Voskresenskaya, E., Ivanitskiy, I., Takhaviev, R., Marchenko, E., Alexandrov, Y.: Smartcheck: Static analysis of ethereum smart contracts. In: 1st

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/s10270-021-00959-x

IEEE/ACM International Workshop on Emerging Trends in Software Engineering for Blockchain, WETSEB@ICSE 2018, Gothenburg, Sweden, May 27 - June 3, 2018. pp. 9–16. ACM (2018), http://ieeexplore.ieee.org/document/8445052
74. Tran, A.B., Lu, Q., Weber, I.: Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management. In: BPM Demos. pp. 56–60 (2018), http://ceur-ws.org/Vol-2196/BPM_2018_paper_12.pdf
75. Viriyasitavat, W., Xu, L.D., Bi, Z., Sapsomboon, A.: Blockchain-based business process management (BPM) framework for service composition in industry 4.0. J. Intell. Manuf. **31**(7), 1737–1748 (2020), https://doi.org/10.1007/s10845-018-1422-y
76. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: BPM. pp. 329–347 (2016), https://doi.org/10.1007/978-3-319-45348-4_19
77. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger (2018), https://ethereum.github.io/yellowpaper/paper.pdf, accessed: 2018-04-07
78. Xu, X., Pautasso, C., Zhu, L., Lu, Q., Weber, I.: A pattern collection for blockchain-based applications. In: Proceedings of the 23rd European Conference on Pattern Languages of Programs, EuroPLoP 2018, Irsee, Germany, July 04-08, 2018. pp. 3:1–3:20. ACM (2018), https://doi.org/10.1145/3282308.3282312
79. Xu, X., Weber, I., Staples, M.: Architecture for Blockchain Applications. Springer (2019), https://doi.org/10.1007/978-3-030-03035-3
80. Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., Pautasso, C., Rimba, P.: A taxonomy of blockchain-based systems for architecture design. In: 2017 IEEE International Conference on Software Architecture, ICSA 2017, Gothenburg, Sweden, April 3-7, 2017. pp. 243–252. IEEE Computer Society (2017), https://doi.org/10.1109/ICSA.2017.33
81. Zhou, X., Jin, Y., Zhang, H., Li, S., Huang, X.: A map of threats to validity of systematic literature reviews in software engineering. In: 23rd Asia-Pacific Software Engineering Conference, APSEC 2016, Hamilton, New Zealand, December 6-9, 2016. pp. 153–160. IEEE Computer Society (2016), https://doi.org/10.1109/APSEC.2016.031

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/s10270-021-00959-x

This document is a pre-print copy of the manuscript
(Di Ciccio, Meroni, and Plebani 2022)
published by Springer (available at link.springer.com).

The final version of the paper is identified by DOI: 10.1007/s10270-021-00959-x

References

Di Ciccio, Claudio, Giovanni Meroni, and Pierluigi Plebani (2022). "On the adoption of blockchain for business process monitoring". In: *Software and Systems Modeling* 21.3, pp. 915–937. ISSN: 1619-1366. DOI: 10.1007/s10270-021-00959-x.

BibTeX

```
@Article{         DiCiccio.etal/SoSyM2022:BlockchainForProcessMonitoring,
  author        = {Di Ciccio, Claudio and Meroni, Giovanni and Plebani,
                   Pierluigi},
  title         = {On the adoption of blockchain for business process
                   monitoring},
  journal       = {Software and Systems Modeling},
  year          = {2022},
  volume        = {21},
  number        = {3},
  pages         = {915-937},
  issn          = {1619-1366},
  doi           = {10.1007/s10270-021-00959-x},
  keywords      = {Blockchain; Distributed ledger technology; Business
                   process management; Software architectures; Business
                   process monitoring},
  publisher     = {Springer}
}
```