Combining Artifact-Driven Monitoring with Blockchain: Analysis and Solutions

Giovanni Meroni(✉) and Pierluigi Plebani

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133 Milano, Italy
{giovanni.meroni,pierluigi.plebani}@polimi.it

Abstract. The adoption of blockchain to enable a trusted monitoring of multi-party business processes is recently gaining a lot of attention, as the absence of a central authority increases the efficiency and the effectiveness of the delivery of monitoring data. At the same time, artifact-driven monitoring has been proposed to create a flexible monitoring platform for multi-party business processes involving an exchange of goods (e.g., in the logistics domain), where the information delivery does not require a central authority but it lacks of sufficient level of trust. The goal of this paper is to analyze the dependencies among these two areas of interests, and to propose two possible monitoring platforms that exploit blockchain to achieve a trusted artifact-driven monitoring solution.

Keywords: Artifact-driven monitoring · Trusted process monitoring
Cyber-physical systems

1 Introduction

To promptly satisfy the ever-changing needs of their customers, organizations must become more flexible and open to changes and opportunities. The *servitization* paradigm [8] goes into this direction: instead of owning corporate assets, be them physical goods, human resources, or business activities, an organization establishes contracts with other organizations, named service providers, that, in exchange for a periodic fee, grant the use of such assets together with value-added services (e.g., maintenance).

As a consequence of servitization, many business processes that were internal now cross the boundaries of single organizations, thus becoming *multi-party*. This also affects the goods that participate in the process, that could be now manipulated and possibly altered by multiple organizations when the process is executed. Additionally, the identity of the service providers involved in these multi-party processes is subject to frequent changes. In fact, short-term contracts with service providers that best fit the needs of a specific process instance are more and more preferred to long-term contracts that span among all the instances of a given business process.

R. Matulevičius and R. Dijkman (Eds.): CAiSE 2018 Workshops, LNBIP 316, pp. 103–114, 2018.
https://doi.org/10.1007/978-3-319-92898-2_8

Despite the previously mentioned advantages, servitization also requires organizations involved in a multi-party process to *trust* each other: i.e., to ensure that the portion of the process assigned to a party is executed as agreed with the other parties and, in case of problems, deviations are correctly reported to properly identify the cause of failures.

In our previous research work, we proposed a novel approach to monitor multi-party business processes, named artifact-driven monitoring [5]. By exploiting the Internet of Things (IoT) paradigm, artifact-driven monitoring makes physical objects *smart*, that is, makes them aware of their own conditions. Based on the conditions of these smart objects, it is then possible to identify when activities are executed and if they are executed as expected, promptly alerting the organizations when a violation occurs. Since smart objects can monitor activities regardless of the organization responsible for their execution, they can take both the orchestration (i.e., the execution of process portions internal to an organization) and the choreography (i.e., the coordination among different organizations) into account.

Although artifact-driven monitoring solves the problem of keeping track of the execution of processes that span across multiple organizations, it does not fully solve the problem of trust among organizations. In particular, the owners of the smart objects are responsible for properly configuring them by specifying which process has to be monitored, based on which physical conditions are activities identified as being executed, and with which other smart objects should monitoring information be exchanged. Therefore, organizations could intentionally misconfigure smart objects in order not to detect violations caused by them.

To guarantee the required level of trust when implementing an artifact-driven monitoring solution, this paper proposes to adopt blockchain technology that, by definition, provides a trusted environment that perfectly suits the needs of multi-party business process monitoring. Moreover, this paper introduces and compares two possible artifact-driven monitoring platforms relying on a permissioned blockchain to propagate monitoring information.

The remainder of this paper is structured as follows: Sect. 2 introduces a motivating example justifying the need for artifact-driven monitoring, which is briefly described in Sect. 3. In Sect. 4 the architecture of the two possible blockchain-based solutions are presented and compared. Finally, Sect. 5 surveys related work and Sect. 6 draws the conclusions of this work and outlines future research plans.

2 Motivating Example

To better understand the importance of a reliable and trusted process monitoring solution, a case study concerning the shipment of dangerous goods is adopted. The actors are a manufacturer M, a customer C, and a truck shipper S that is involved when potentially explosive chemicals must be delivered from M to C.

The delivery process is organized according to the Business Process Model and Notation (BPMN) model shown in Fig. 1. To avoid possible accidents during

the execution of this process, the following countermeasures must be put in place. Firstly, if a leakage of the chemical is detected when M is filling a tank, no matter how modest, M must empty and replace the tank. Secondly, if the tank gets overheated while it is being shipped, S must immediately abort the shipment, notify the authorities about this potential hazardous occurrence, and wait for them to intervene.

Like in every multi-party business process, also in this case each organization is in charge only of the activities included in their pools, thus nobody has full control on the whole process. Consequently, being able to identify when activities are performed and if they are performed as expected is required to determine who caused the process not to be executed as expected. This becomes particularly important if an accident occurs, as depending on which portion of the process was not executed as expected, the organization to blame for the accident varies. For example, if the chemical explodes while the tank is being unloaded from the truck, it could be determined by a multitude of causes. Firstly, the tank may have been overheated and S decided to ignore that event and go on shipping it. Alternatively, the tank may have had a leakage and have not been replaced by M.

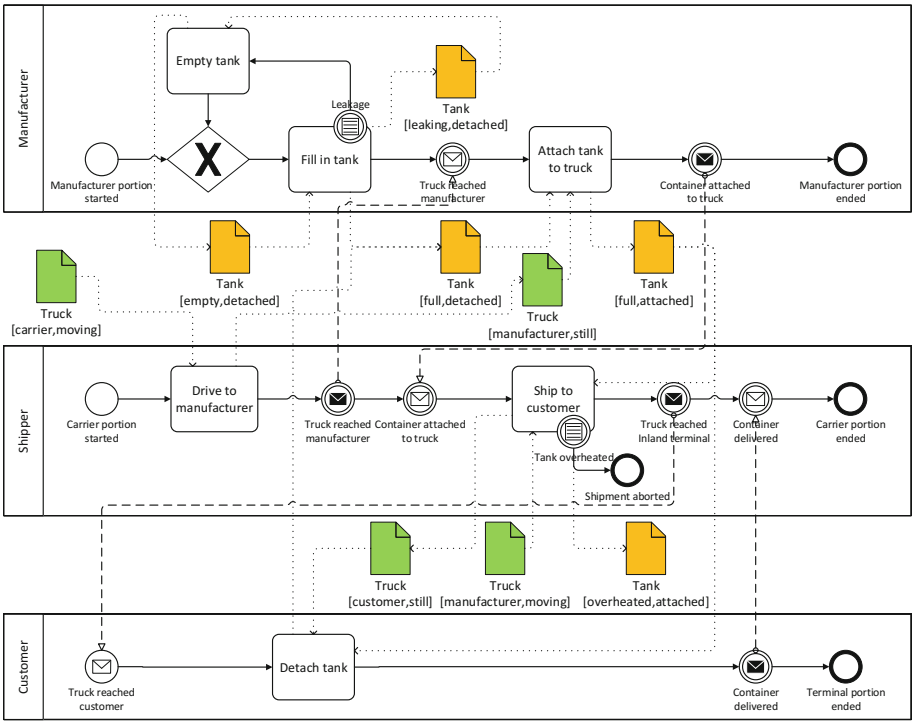

Fig. 1. BPMN diagram of the motivating example.

3 Artifact-Driven Monitoring

Nowadays, several organizations rely on a Business Process Management System (BPMS) to manage their processes, as it enacts the execution of business activities, monitors the process, and identifies potential issues. When a process includes human-based activities, operators are in charge of explicitly notifying the activation and completion of these activities. This way, the BPMS has knowledge about if, and when, those activities are performed. Consequently, the organizations have to trust the operators to provide timely notifications that reflect the actual execution of the process. In addition, when a multi-party process takes place, organizations are required to federate their own BPMS to be able to monitor the whole process. Alternatively, a centralized BPMS must be put in place, and fed with notifications sent by all the participating organizations. In both cases, relevant system integration efforts must be undertaken, and each organization have to trust the other ones.

To solve these issues, artifact-driven monitoring [5] relies on a completely different approach. Instead of requiring organizations to actively monitor a process, it moves the monitoring tasks onto the physical objects (i.e., artifacts) participating in the process. To this aim, activities operating on those objects should be defined in terms of pre-conditions (i.e., status of the objects before the activity can be executed) and post-conditions (i.e., status of the objects determining the completion of the activity). For example, as shown in Fig. 1, to execute activity *Attach tank to truck* the tank must be full and detached from the truck, and the truck must be located at the manufacturer plant and stay still. Similarly, we assume *Attach tank to truck* to be completed when the tank is full and attached to the truck.

Therefore, as long as those physical objects *(i)* are aware of the process being performed, *(ii)* can autonomously determine their own conditions, and *(iii)* can exchange this information with the other objects, they can passively monitor the process without requiring human intervention. This is made possible thank to the IoT paradigm, that makes physical objects smart, that is, equipped with sensors, a computing device, and a communication interface.

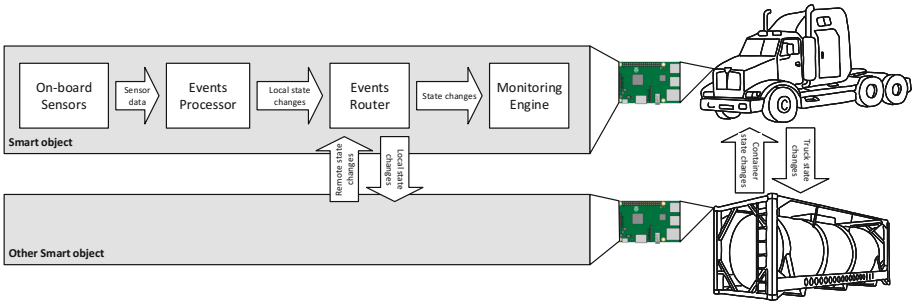

Fig. 2. Architecture of an artifact-driven monitoring platform.

Figure 2 shows the current architecture of an artifact-driven monitoring platform [5]. Firstly, the physical characteristics of a smart object are captured by *On-board Sensors*. Then, sensor data are processed and discretized by the *Events Processor* module, in order to assign to the smart object a single state from a finite set, representing its current conditions according to the possible states defined in the BPMN process model (e.g., empty, leaking). Whenever a change in the state of the smart object is detected by the *Events Processor*, this information is forwarded by the *Events Router* to the other smart objects participating in the process. As soon as the *Events Router* receives a change in the state of a smart object, either local or remote, it forwards it to the *Monitoring Engine* module, that uses this information to actively monitor the process. To do so, the *Monitoring Engine* relies on an Extended-GSM (E-GSM) model[1], which contains a formal representation of the process agreed among the organizations, enriched with information on the physical objects and states required by and altered by the business activities composing the process. This way, whenever the *Monitoring Engine* determines that an activity is being carried out or that the process is not executed as agreed, it can promptly inform the organizations participating in the process.

4 Approach

Artifact-driven monitoring allows to continuously and autonomously monitor multi-party business processes. However, it does not solve the problem of trust among organizations. In fact, organizations are in charge of configuring their own smart objects with the process to monitor, and deliberately erroneous configuration could be not detected.

For example, still referring to the accident described in Sect. 2, if the tank detects that it was overheated while being shipped, S could argue that M (the owner of the tank) incorrectly implemented the monitoring platform, or worse, it intentionally omitted, from the agreed process model, to monitor the portion responsible for detecting a leakage. In this case, an independent authority would typically have to investigate on the actual cause of such an accident, and on the correct implementation and configuration of the monitoring platform. However, this task could be really difficult. For example, the explosion may have completely destroyed the smart objects, so the authority could no longer rely on monitoring information to determine the real cause.

To solve these issues, we propose to combine artifact-driven process monitoring with blockchain. A blockchain provides a shared immutable ledger, which guarantees that each entry *(i)* has to be validated before being stored, *(ii)* is persistently stored, *(iii)* is replicated along multiple nodes, and *(iv)* is immutable. This way, by storing monitoring information on a blockchain, such an information can be accessed and validated by all the participants of a process both during its execution and after it is completed. In addition, independent auditors

[1] We refer you to [5] to understand the advantages of E-GSM over other process modeling languages when doing runtime process monitoring.

can also access and validate monitoring information even if the smart objects are no longer accessible, as the needed information has been distributed to the several ledgers composing the blockchain environment.

Sections 4.1 and 4.2 present two possible architectures of an artifact-driven monitoring solution integrating a blockchain, discussing the advantages and disadvantages of each solution.

Regardless of the specific approach, the adoption of a blockchain protocol to support the artifact-driven monitoring approach has the advantage of providing a native, trusted, and reliable environment for sharing monitoring information among the parties. Moreover, we propose the adoption of a permissioned blockchain [12] instead of a public one, such as Ethereum[2]. A public blockchain allows anyone to read and write entries to it, requiring the participants to manually manage the encryption of entries in case they want them to be kept confidential. Instead, a permissioned blockchain natively supports access control mechanisms to ensure that only authorized participant can read and/or write blocks on the blockchain. This way, only organizations and smart objects participating in the process can be granted write access. Also, read access can be restricted in case the participants of a multi-party process do not want information on the execution of the process be available to anyone. Moreover, adopting Ethereum as blockchain protocol gives also the possibility to define the so-called *smart contracts*, that can be defined as functions, coded using languages like Solidity[3], whose execution determines either the acceptance or the refusal of a block to be written in the chain. Since the blocks to be written concerns the status of the smart objects, we assume that smart contracts could be inferred from the initial process model. In fact, once a process instance is created (and the corresponding monitoring infrastructure properly configured), the process specification contains all the information needed to determine under which conditions smart objects change their state.

As one of the main drawbacks of blockchain concerns the performances, especially in terms of throughput and latency, both architectures adopts multiple chains to reduce space, bandwidth and computational requirements [11]. In order to work, a blockchain requires all the participants to keep track of all the entries that were written to it. Therefore, if a single chain is used by all the organizations implementing our artifact-driven monitoring solution, each smart object and organization would have to also store entries that are completely unrelated to the process they are currently monitoring. This causes the hardware and network requirements to grow proportionally to the number of entries that are written to the blockchain, and it is one of the issues that limit the scalability of a blockchain. To mitigate this issue, multiple chains based on the same blockchain protocol can be deployed. In particular, before a new execution of a process takes place, we propose to deploy a new chain and instruct the smart objects and organizations participating in that specific execution to use that chain. By doing so, each chain contains only entries that are specific to the current process

[2] See https://www.ethereum.org.
[3] See https://solidity.readthedocs.io/en/develop/.

execution, thus sensibly limiting the hardware and network requirements of the smart objects.

Focusing on the definition of smart contract, we propose to add a block to the chain only when a smart device detects a change in its state, and we envision two alternatives concerning the content of a block. In the first case, the block contains only the new state. Consequently, the smart contract only checks if the smart object adding the block is the one whose state changed (see Sect. 4.1). In the second case, the block contains both the new state and the series of sensor data causing the smart object to detect the change of state. In this case, the smart contract has to encode rules to verify that the change of state is compatible with the sensor data (see Sect. 4.2). As the information to be stored in a block is generated by different elements of the artifact-driven monitoring platform, these two alternatives have an impact on the configuration of the platform.

4.1 State-Oriented Block

A first approach is named state-oriented blockchain, as a new block is added whenever a smart object realizes that its state has changed.

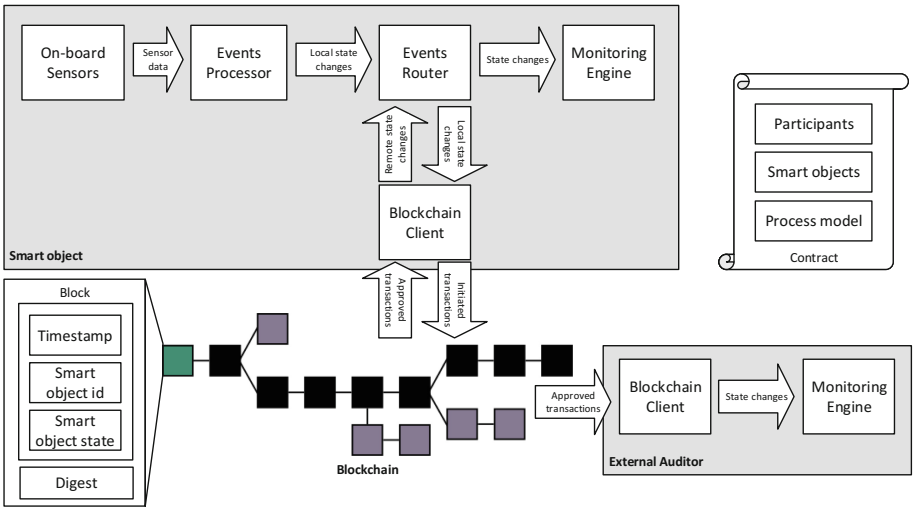

Fig. 3. State-oriented blockchain.

Figure 3 shows the reference architecture, where the blockchain is used to let smart objects exchange information on their current state. Instead of directly communicating with the other smart objects, the *Events Router* module relies on a new module, named *Blockchain Client*. The *Blockchain Client* is responsible for writing a new block whenever the current smart object changes state, and for sending a notification to the *Events Router* whenever a new block is written by the other smart objects.

Before the process starts, the identity of the organizations participating in the process and of the smart objects owned by them is extracted from the E-GSM model and formalized in a smart contract, which is written to the blockchain and approved by all the participants. The E-GSM model is also included in the description of the smart contract, to make all participants, including smart objects and external auditors, aware of the process they have to monitor.

The payload of each block in the blockchain contains a timestamp, the identity of the smart object producing the block, and the current state of the smart object. To ensure that the block was produced by the smart object it refers to, a Public Key Infrastructure (PKI) is put in place. Each smart object owns a private key, which is used to sign the block and kept private, and a public key, which is distributed among all the participants in the process. This way, thank to the PKI and the smart contract, each node approves the block only if *(i)* the signature matches the payload, and *(ii)* the smart object writing the block is actually owned by the participating organizations. Since a permissioned blockchain is adopted and, consequently, the identity of the participants is known, there is no need to put in place computationally expensive consensus algorithms, such as proof of work. Instead, simpler algorithms, such as Practical Byzantine Fault Tolerance (PBFT), are sufficient.

This architecture presents several advantages. Firstly, external auditors can easily monitor the process, either while it is being executed or after it completed. Thank to smart contracts, they can be sure that changes in the state of the smart objects were written to the blockchain only if they originated from the same smart object. In addition, as the E-GSM model is enclosed in the description of the smart contract, they can trust such a model and be certain that it represents the process approved by all the organizations. Then, by instructing a *Monitoring Engine* with the E-GSM model and feeding it with changes in the state of the smart objects extracted from the blocks in the blockchain, auditors can independently verify if the process was executed as agreed. For example, still referring to the example described in Sect. 2, if this architecture is implemented by M, S and C, the authorities can easily identify the organization responsible for the accident, even if the smart objects were destroyed. In fact, authorities can simply query the blockchain to obtain the E-GSM process and all the changes in the state of the smart objects, being sure that this information was not altered once it was written to the blockchain. Then, they can instruct a *Monitoring Engine* with the E-GSM model, and replay the state changes to detect which portion of the process was incorrectly executed.

However, this architecture also presents one limitation. By design, the blockchain does not store information on how to determine from sensor data the state of the smart objects. Consequently, organizations could argue that smart objects incorrectly determined their state, thus providing unreliable monitoring data. For example, when instructing the monitoring platform running on its tanks, M could have intentionally left out information on how to determine when the tank is leaking. Therefore, even if a leakage occurs, the monitoring platform would never be able to detect and write to the blockchain this information.

As a consequence, based only on the information on the blockchain, external authorities would not be able to notice that the process was not executed as expected.

4.2 Sensor-Oriented Block

To address the limitation of the previous approach, a new one, named sensor-oriented blockchain, is introduced. This approach also makes use of smart contracts. However, besides formalizing the organizations participating in the process and the identity of the smart objects, each smart contract also defines the rules to detect from sensor data when the smart objects change state. This makes it possible for all participants to agree on how the state of the smart objects should be detected. For example, to detect when a truck assumes the state *moving*, all participants will require the GPS coordinates of the truck to vary of at least 0.01 degrees per minute.

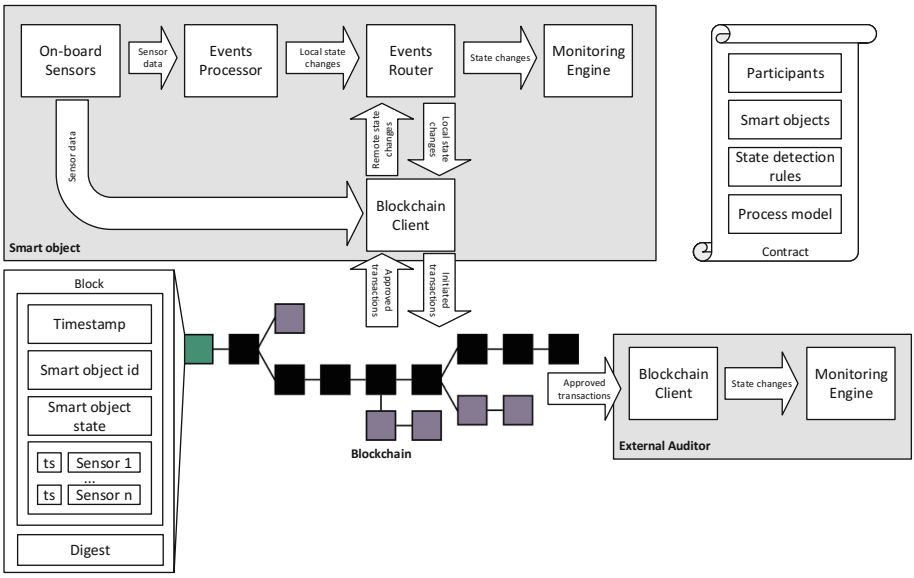

Fig. 4. Sensor-oriented blockchain.

To support this approach, the reference architecture shown in Fig. 4 is proposed. In this case, the *Blockchain Client* also receives data from the *On-board Sensors*, and encloses this information to the blocks written to the blockchain. Like in the previous architecture, a new block is written to the blockchain whenever the smart object assumes a new state. However, besides containing a timestamp, the identity of the smart object producing the block, and the current state of the smart object, each block also contains a list of sensor values, together with

a timestamp, that were collected since when the previous state was detected. For example, a block written by the truck when it transitions from *moving* to *still* will contain the state *still*, the date and time when *still* was detected, the identifier of the truck, and all the GPS coordinates that the truck assumed since when the state *moving* was previously detected. Also in this case, the architecture relies on a PKI infrastructure to ensure that the block was produced by the smart object it refers to.

With respect to the previous approach, this one achieves an even greater level of trust. In fact, thank to the smart contract, every participant can autonomously verify if the state detected by a smart object is reflected by the sensor data and, if not, discard the corresponding block. Since the rules to determine this information are defined in the smart contract, and are approved by all organizations before the process takes place, nobody can argue that the process is incorrectly monitored.

However, this approach also presents some issues that are not present in the previous one. Firstly, it makes a much more intensive use of the blockchain. In fact, the size of each block can be quite large, as it also contains a set of sensor data. Therefore, depending on the sampling rate of the sensors and on the duration of the process, the blockchain can grow significantly while the process is being monitored. Nevertheless, thank to the typically small size of sensor data, storage requirements are still quite modest. However, network requirements, especially in terms of bandwidth, can grow significantly.

Another issue of this approach is the higher workload that each smart object must handle. In fact, besides determining the state of its own smart object, each smart object also has to verify if the state indicated by the other smart objects is correct. Thus, the computational requirements of the architecture depend on the total number of rules defined in the smart contract, and on their complexity. To compensate for this issue, additional nodes deputed to the verification of the blocks can be deployed on the organizations' premises, thus freeing smart objects from this task.

5 Related Work

Given the recent affirmation of blockchain in many domains, the Business Process Management (BPM) research community is also investigating the impact of this technology on all the phases of the BPM lifecycle. In [4] an exhaustive analysis of the implications of introducing blockchain in inter-organizational processes are discussed, and a list of seven possible future research directions is identified. Among them, the goal of this paper is mainly "developing a diverse set of execution and monitoring frameworks on blockchain", albeit the proposed solution may also affect other directions. To this aim, particular emphasis is given on a solution that is able to deal with some of the main aspects, namely throughput and size, that could hamper the adoption of blockchain in BPM. In fact, the proposed solution distributes the workload to several chains, resulting in a reduced amount of transactions per single chain, with consequent low hardware

and network requirements. In addition, the two alternatives discussed in the paper further decrease the computational effort to be done on the blockchain by moving some of the computation to off-chain, as also suggested in [2].

Focusing on the usage of blockchain to monitor a supply chain, the literature is currently investigating this issue according to different perspectives. For instance, [3] analyses the possible grind between a blockhain infrastructure with the information management infrastructure currently adopted in the shipping domain. Moreover, in [10] a framework for process monitoring based on several private blockchain installations, globally managed by a public blockchain, is presented. In some way, the idea of having several chains to increase the confidentiality of the information among the stakeholders is similar to what it is proposed in this paper. Nevertheless, [10] considers monitoring as a centralized element, while in our approach it is distributed among the smart objects.

An interesting report, [7], proposes two different approaches: the first one relies on a common blockchain to collect all the events coming from the different stakeholders involved in the supply chain, while the second one is based on the usage of smart contracts. In both cases, the approach assumes to start from the complete definition of the choreography to configure the blockchain. As discussed in the paper, this introduces a significant problem related to the encryption of the data stored in the ledgers, as not all the information can be read by all the participants in the blockchain. In our approach, each process execution relies on a specific chain, which is accessible only by the organizations participating in that specific process execution. This solves the problem of the data confidentiality, as the data stored in one of the blockchains should be visible by all the participants. Finally, [6] proposes an interesting solution for run-time verification of business process execution based on Bitcoin, thus, with a public and very specific solution where smart contracts are not allowed.

6 Conclusions and Future Work

This paper presented how artifact-driven monitoring can benefit from blockchain to monitor multi-party processes in a trusted way. Thank to a permissioned blockchain, monitoring information is stored immutably and persistently, allowing external auditors to independently verify if the process was performed as expected, either at runtime or after the process completed.

One of the disadvantages of our approach concerns the initial set-up, as having several blockchains requires the configuration of all of them. To solve this limitation, we plan to adopt the approach proposed in [9] for configuring a blockchain-based solution starting from the choreography model.

Another potential disadvantage of this approach consists in the limited speed of blockchain. In fact, writing, approving, and distributing a new block to all the participants takes seconds for a permissioned blockchain, or even several minutes for a public one. Nevertheless, research efforts to speed up operations on a blockchain are currently being taken by both academics and the industry, so we expect this issue to be eventually solved or scaled back.

Our future research work will consist in implementing a prototype of both architectures, and validating it with real-world processes and sensor data. In addition, we will also consider the introduction of side-chains [1] to allow smart objects to monitor multiple processes at the same time, and to integrate process monitoring with automatic payment and escrow mechanisms. Finally, being the artifact-driven monitoring a nomadic infrastructure, the impact of the lack of connectivity in a blockchain solution will be investigated.

Acknowledgments. This work has been partially funded by the Italian Project ITS Italy 2020 under the Technological National Clusters program.

References

1. Croman, K., et al.: On scaling decentralized blockchains - (A position paper). In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 106–125. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_8
2. Eberhardt, J., Tai, S.: On or off the blockchain? Insights on off-chaining computation and data. In: De Paoli, F., Schulte, S., Broch Johnsen, E. (eds.) ESOCC 2017. LNCS, vol. 10465, pp. 3–15. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67262-5_1
3. Jabbar, K., Bjørn, P.: Infrastructural grind: introducing blockchain technology in the shipping domain. In: GROUP 2018, pp. 297–308. ACM, New York (2018)
4. Mendling, J., et al.: Blockchains for business process management - challenges and opportunities. ACM Trans. Manage. Inf. Syst. **9**(1), 4:1–4:16 (2018)
5. Meroni, G., Baresi, L., Montali, M., Plebani, P.: Multi-party business process compliance monitoring through IoT-enabled artifacts. Inf. Syst. **73**, 61–78 (2018)
6. Prybila, C., Schulte, S., Hochreiner, C., Weber, I.: Runtime verification for business processes utilizing the Bitcoin blockchain. Future Gener. Comput. Syst. (2017)
7. Staples, M., Chen, S., Falamaki, S., Ponomarev, A., Rimba, P., Tran, A.B., Weber, I., Xu, X., Zhu, J.: Risks and opportunities for systems using blockchain and smart contracts. Technical report, Data61 (CSIRO) (2017)
8. Vandermerwe, S., Rada, J.: Servitization of business: adding value by adding services. Eur. Manage. J. **6**(4), 314–324 (1988)
9. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 329–347. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_19
10. Wu, H., Li, Z., King, B., Ben Miled, Z., Wassick, J., Tazelaar, J.: A distributed ledger for supply chain physical distribution visibility. Information **8**(4), 137 (2017)
11. Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A.B., Chen, S.: The blockchain as a software connector. In: WICSA 2016, pp. 182–191. IEEE (2016)
12. 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: ICSA 2017, pp. 243–252. IEEE (2017)