Extracting Event Logs for Process Mining from Data Stored on the Blockchain

Roman Mühlberger[1], Stefan Bachhofner[1], Claudio Di Ciccio[1], Luciano García-Bañuelos[2,3], and Orlenys López-Pintado[2]

[1] Vienna University of Economics and Business, Austria
roman@muehlberger.eu.com, {stefan.bachhofner,claudio.di.ciccio}@wu.ac.at
[2] University of Tartu, Estonia
orlenyslp@ut.ee
[3] Tecnológico de Monterrey, Mexico
luciano.garcia@tec.mx

Abstract. The integration of business process management with blockchains across organisational borders provides a means to establish transparency of execution and auditing capabilities. To enable process analytics, though, non-trivial extraction and transformation tasks are necessary on the raw data stored in the ledger. In this paper, we describe our approach to retrieve process data from an Ethereum blockchain ledger and subsequently convert those data into an event log formatted according to the IEEE Extensible Event Stream (XES) standard. We show a proof-of-concept software artefact and its application on a data set produced by the smart contracts of a process execution engine stored on the public Ethereum blockchain network.

Keywords: Ethereum·Process Discovery·Process Monitoring·Process Conformance

1 Introduction

Blockchain offers a new prospective environment for the execution of inter-organisational processes [24]. On top of a distributed execution environment, blockchains collate transactions onto a push-only stack (ledgers) within backward-linked blocks [25]. Participants are provided with an up-to-date version of the entire blockchain via broadcast after a new block has been added following a consensus algorithm. The utilisation of the blockchain in processes covering a number of participants prevents the necessity of a central orchestrator and provides trust among mutually untrusted parties [31]. Additionally, the decentralised database stores transactional information triggered by smart contracts containing the state and execution details of distinct cases [24]. Data can be retrieved that contains, inter alia, interacting process participants, (discrete) time and duration of the interaction, execution costs on the blockchain as well as the state details of a limited, recurring set of activities. This information enables process monitoring [7], process compliance [32], and process conformance checking [23,26] via process mining [29,30]. In addition, the information is also valuable for the analysis of the efficiency and effectiveness of process executions, i.e., for performance mining [15,11,19,4].

In order to make proper use of that information, manual effort is required to convert the data from individual blocks into an appropriate format, which causes a variety of challenges [27]. To mention but a few, the information retrieved from the blockchain is represented in hexadecimal and numeric formats; timestamps are approximate as the finest granular unit of time in blockchains is at the level of blocks; the structure of data payloads attached to transactions is for the most part arbitrary. Process mining tools, on the other hand, present clear constraints on data types and their required representation. In order to localise and convert the data, a profound understanding of the data model represented on the blockchain is essential.

In this work-in-progress paper, we follow the Design Science research methodology to devise a framework that retrieves process data from blocks and transforms the process data into an event log, formatted according to the IEEE Extensible Event Stream (XES) standard. The software artefact is reflected in an implemented prototype, which is applied on a real-world case study. The XES output file is used as an input for existing tools in the *ProM* toolkit. Additionally, we discuss the challenges and opportunities pertaining to data extraction, transformation and interpretation of blockchain data to drive future work in this area. To the best of our knowledge, this work is the first one retrieving and transforming blockchain data into a standardised format that can be used in process mining tools. Consequently, our work is the first that enables process monitoring, process conformance and process compliance checking with process mining on blockchain-based process executions.

The remainder of this paper is structured as follows. In Section 2, the notions backing our research work are provided. Section 3 describes our approach. Section 4 describes the experiment conducted with a real-world case study. The paper concludes with Section 5 including implications for further research.

2 Background

A blockchain consists of a distributed ledger of transactions [25]. Blockchains operate on top of a peer-to-peer network, where each peer stores a local copy of the ledger's transactional data. The transactions are collated in blocks, which are sequentialised as an append-only chain. Compact identification and storage of data, as well as the backward-linking of blocks in the chain, are based upon hashing algorithms such as the SHA-3 compliant algorithm KECCAK [6]. Appending new blocks in a trustworthy manner is enabled by a combination of consensus-making, cryptography, and market mechanisms [25]. A central authority is thus not necessary to that extent, making it advantageous for processes containing a set of untrusted and unknown participants [31]. More recent blockchain protocols such as Ethereum [33] enable programmability of the platform through the *smart contracts*. Ethereum distributions come endowed with dedicated smart-contract programming languages such as Solidity [10], an object-oriented, high-level programming language. Ethereum smart contracts are executed in a decentralised environment named Ethereum Virtual Machine (EVM). Their function signatures and public attributes are exposed through a humand- and machine-readable interface called Application Binary Interface (ABI). The execution of smart contract operations is associated to a price expressed in terms of *gas*, exchanged at a variable rate with the Ether cryptocurrency. The invocation of smart contracts' functions occurs via transactions that are stored on the ledger.

Blockchain for business process execution. Smart contracts allow for the codification of business process logic on the blockchain [12], as shown in the seminal work of Weber et al. [31] and later on with tools like Caterpillar [22] and Lorikeet [28]. As several modern Business Process Management Systems (BPMSs) do, Caterpillar and Lorikeet adopt a

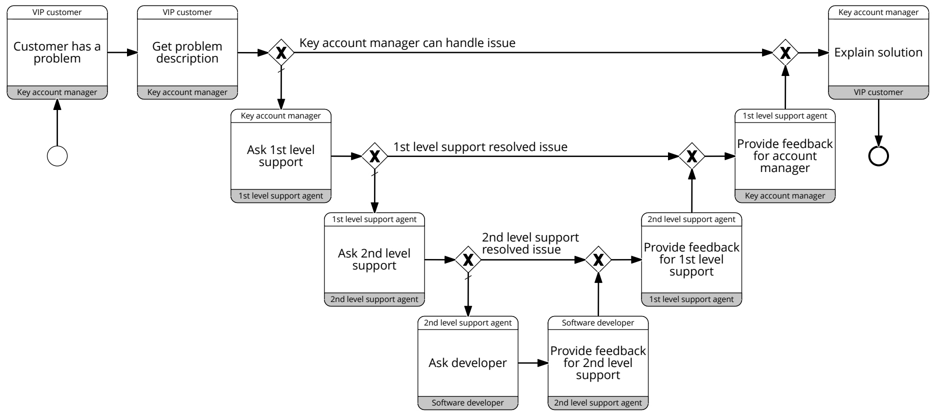


Fig. 1: The choreography Business Process Model and Notation (BPMN) model of an incident management process [31].

Model-Driven Engineering (MDE) approach to let the process analysts provide graphical representations of the process and turn it into executable code enacting it. They take as input models that are specified in the BPMN language [15]. Figure 1, e.g., illustrates the BPMN choreography model of the incident management process introduced by Weber et al. [31] to evaluate their execution engine for inter-organisational processes.[4] The process begins with an issue raised by a customer. The key account manager registers the problem description. Until a solution is not found, the ticket escalates to the 1st level support, then to the 2nd level support, and finally to the developer. After the solution is found, a feedback is given and the solution explained to the customer.

The workflow routing of blockchain-based process execution engines is performed by smart contracts generated by compilers that translate BPMN diagrams into smart contract code (e.g., Solidity). Once deployed on the blockchain, the contract instances are identified by the hexadecimal address of their account, which serve as a key for querying the state of the process instances.

The transactional storage of interactions among the actors operating on the blockchain and with smart contracts enables traceability of processes [13]. Our aim is to extract the process data from the ledger and turn it into a readily processable format for process mining tools, in order to enable analysis and auditing of processes run on the blokchain.

Event logs for process mining. Process mining makes use of the sequential data stored by BPMS and other process-aware information systems such as Enterprise Resource Plannings (ERPs) or Customer Relationship Managements (CRMs) in order to discover, analyse and enhance existing processes [1]. As such, process mining is a valuable means to conduct auditing and forensics on existing process data [18]. The standard data structure with which those data are stored is named *event log*, or log for short. An event log consists of a collection of traces, where every *trace* represents a process run (or process instance) and activity executions

[4] Diagram in Fig. 1 courtesy of Ingo Weber.

Table 1: An event log excerpt.

Case ID	Event ID	Activity name	Timestamp	Resource	…
1	1	Customer has a problem	2016-03-22T14:06:22.000Z	Key Account Manager	…
1	2	Get problem description	2016-03-22T14:06:35.000Z	Key Account Manager	…
1	3	Ask 1st level support	2016-03-22T14:07:20.000Z	Key Account Manager	…
1	4	Ask 2nd level support	2016-03-22T14:08:38.000Z	1st Level Support	…
1	5	Provide feedback for 1st level support	2016-03-22T14:09:04.000Z	2nd Level Support	…
1	6	Provide feedback for account manager	2016-03-22T14:09:14.000Z	1st Level Support	…
1	7	Explain solution	2016-03-22T14:09:22.000Z	Key Account Manager	…
2	8	Customer has a problem	2016-03-22T14:37:45.000Z	Key Account Manager	…
2	9	Get problem description	2016-03-22T14:38:00.000Z	Key Account Manager	…
…	…	…	…	…	
32	250	Customer has a problem	2016-03-22T12:46:01.000Z	Key Account Manager	…
32	251	Get problem description	2016-03-22T12:46:21.000Z	Key Account Manager	…
32	252	Ask 1st level support	2016-03-22T12:47:16.000Z	Key Account Manager	…
32	253	Ask 2nd level support	2016-03-22T12:47:53.000Z	1st Level Support	…
32	254	Provide feedback for 1st level support	2016-03-22T12:48:09.000Z	2nd Level Support	…
32	255	Provide feedback for account manager	2016-03-22T12:48:40.000Z	1st Level Support	…
32	256	Explain solution	2016-03-22T12:49:17.000Z	Key Account Manager	…

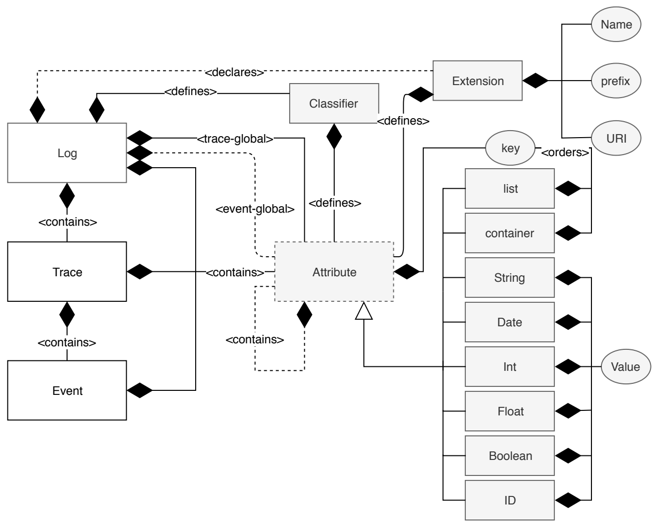


Fig. 2: The meta model class diagram representation of the XES standard [17].

are represented by *events* (each mapped to an activity name). Event logs may contain additional information in the form of event attributes, such as the timestamp or the resource carrying out the related activity. Table 1 shows an excerpt of event log that may originate from the execution of the process in Fig. 1. Process analytics toolkits such as ProM [2] or Apromore [20] take as input event logs that are formatted according to the XML-based IEEE standard XES [17]. Figure 2 illustrates the original version of the standard. The top level element of XES is a log containing trace nodes (one per process instance). Traces incorporate event nodes. Data are stored in *attributes*, which are represented as key-value nodes. The XES standard defines attribute data types and semantics through an extension mechanism. Every extension has a name, a prefix, and a reference Uniform Resource Identifier (URI). For example, extension Concept, with prefix concept:, and URI `http://www.xes-standard.org/concept.xesext`, is used to name elements such as event (typically with the corresponding activity label) and traces. Other

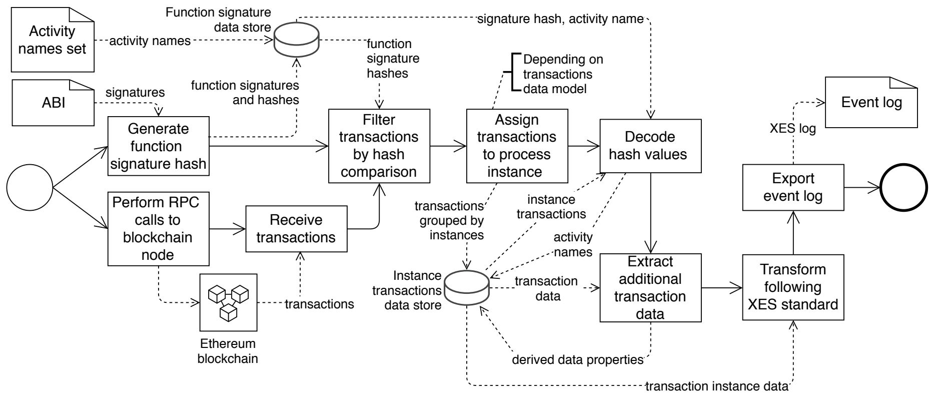

Activity names set
ABI
Function signature data store
activity names
signatures
function signatures and hashes
function signature hashes
signature hash, activity name
Depending on transactions data model
Event log
XES log
Generate function signature hash
Filter transactions by hash comparison
Assign transactions to process instance
Decode hash values
Export event log
Perform RPC calls to blockchain node
Receive transactions
transactions grouped by instances
instance transactions
activity names
Instance transactions data store
transaction data
Extract additional transaction data
Transform following XES standard
transactions
Ethereum blockchain
derived data properties
transaction instance data

Fig. 3: Approach for the extraction of process data from a blockchain and transformation into an event log.

extensions we make use of are, e.g., identity, to denote the unique identifiers of elements, organizational (prefix: org) for resources, time for event time-stamping, lifecycle for the status of activities, and cost for execution costs.

3 Approach

Figure 3 illustrates our approach for the extraction and transformation to XES of process data from the blockchain. We remark that we use only information that is stored on the blockchain.

We begin with the analysis of the smart contract implementing the business process. With this analysis, we aim at producing information to identify transactions that are associated with the execution of process tasks. To that end, we rely on the fact that the payload of an Ethereum transaction stores the *function selector*, i.e., the first four bytes in the SHA-3 hash of a string encoding the function signature. For instance, let us consider task *Customer has a problem*, which is implemented by a function with signature `Customer_Has_a_Problem()`. The selector for such function is `0xefe73dcb`, because the SHA-3 hash for `Customer_Has_a_Problem()` is `0xefe73dcb348c11a7ab31ce1620102e63c94e84ab393a78f187d1485c8a2c72cc`. For convenience, we keep a hash-table to associate task names with their corresponding function selectors, as we use it to generate and enrich the event log.

Blockchain-based BPMSs such as Caterpillar and Lorikeet use the factory pattern to control the creation of process instances: a factory contract deploys, for each new process instance, a smart contract that enacts the business process. We observe that the address of the factory contract is included in the metadata of the transaction associated with the creation of a new process instance as the originator of such transaction, i.e., within the attribute `Transaction.from`. Therefore, if we assume that the address of the factory contract is provided, we can identify the address of each of the process instances created by the factory contract and later each one of the transactions associated with a process instance. We inspect the blockchain data model of blocks and transactions as illustrated in Fig. 4. We identify the hash as a unique identifier for every block and every transaction.

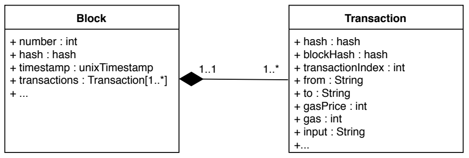


Fig. 4: The data model of blocks and transactions. We depicted the attributes used for the extraction of blockchain data.

The transactions included in a block can be obtained in the Block.transactions field of a block. The UNIX timestamp represented in the block can be used as a basis to derive the timestamps of the activities therein. The transactions are allocated to an enumeration denoting the order of the process executions and saved into a data structure. In order to provide completeness of the dataset, we decode the hash values and save both the function signatures and activity names in a human-readable format. In order to add the activity name, we compare the data from the transaction input field with the signature hash saved in the data structure, which is denoted as *Function signature data store* in Fig. 3. For consistency reasons, we use the transaction data from the data structure denoted as *Instance transactions data store* to add additional information such as the costs of the process instances in Ether or fiat currency like US dollars. Finally, the data extraction and processing is finalised and the result can be transformed into an event log that is compliant with the XES format.

To transform the instance transactions data store into a log that complies with XES standard, we map the attributes in the data store to XES event and trace attributes. Whenever suitable, we append XES extension declarations to our file to enrich events with additional information, such as cost or time, declared into the header of the XES document. For example, we use the mentioned transaction hash `0x656252f3ecee102d981520ca9e0ca0f7048bce99e4f6fead89d358cdbedd6156` as the id node of the event. We use the function selector, i.e., the first four bytes of the Transaction.input field (`0xefe73dcb`) as a custom *activity_id* attribute. Timestamp *2016-03-22T13:44:22.000Z* is mapped to the event with the key time:timestamp.

4 Case Study

In this section, we present a case study for our approach. We extract process data from a public blockchain and transform the process data into a XES log. The software project, including the Jupyter notebook for our case study,[5] is openly available as a public GitLab repository.[6]

4.1 Extraction of an event log from the public Ethereum blockchain

To evaluate our approach, we analyse the transactions stored on the public Ethereum network by the execution engine described by Weber et al. [31]. Figure 1 illustrates the BPMN choreography model of the incident management process they implemented. The process is enacted by

[5] https://gitlab.com/MacOS/extracting-event-logs-from-process-data-on-the-blockchain/tree/paper/incident_management_process.ipynb

[6] https://gitlab.com/MacOS/extracting-event-logs-from-process-data-on-the-blockchain/tree/paper

Table 2: Mapping between event log elements and blockchain data fields in our case study.

Event log attribute	Blockchain data field
Case ID	Transaction.to
Activity ID	Transaction.input[2:10] (function selector)
Event ID	Transaction.hash
Activity label	Reverse-engineered from contract ABI and function selector in Transaction.input
Event timestamp	Block.timestamp (plus sorting by Transaction.index)
Event cost	Transaction.gas
Event resource	Transaction.from

```
{   blockHash:        0x1eca7ae74de59dff4f553b052a0e8346b1fc1587cf76ca5ee5b22da84f87822b,
    blockNumber:      1196772,
    from:             0x1387e74982055e3e1d235aad579350813b329b2b,
    gas:              1000000,
    gasPrice:         20000000000,
    hash:             0x656252f3ecee102d981520ca9e0ca0f7048bce99e4f6fead89d358cdbedd6156,
    input:            0xefe73dcb,
    nonce:            227,
    r:                0xf26831a097ee1a3cf64364a07ff80fa816dd8604461482921a81be74276b5e7b,
    s:                0x60a41b999ccd10461565d604cd063c299a5b1006a9ed8b0fcc84e5cfa9960f8d,
    to:               0x0E6e0313dBe1Ba7A8bCb622EE7A77EaCBc9eF73f,
    transactionIndex: 3,
    v:                28,
    value:            0 }
```

Listing 1: Key-value representation of the transaction data denoting the execution of an activity.

a smart contract to be run on the blockchain, which we henceforth refer to as process contract. Every activity corresponds to a function, whose signature is exposed by the contract ABI.

The process instances implemented on the blockchain are initiated by a factory smart contract at address `0x09890f52cdd5d0743c7d13abe481e705a2706384`. The factory contract deploys a new instance of process contract for every new run. To retrieve the process data, we access the Etherscan Ethereum blockchain explorer and analyse the list of transactions having the factory smart contract as the sender, i.e., such that the factory address occurs in their Transaction.from field. Those transactions amount to 32 in our case.[7]

Table 2 summarises the mapping between the event log elements and the blockchain data. The Transaction.to field of each of the transactions from the factory contract identifies a process instance, thus 32 process instances were run on the blockchain. Using again Etherscan, we retrieve the transactions directed to every such process instance contract. Listing 1 shows one of those transactions. To identify which of those correspond to an activity enactment, we operate as follows. Considering the set of activities of the process, we find the corresponding function names in the ABI of the called contract. Thereupon, we compute the KECCAK hash for each of the function signatures of interest and save the hexadecimal representation of its first four bytes, as shown in Table 3. We thus select the transactions denoting the activity enactment by selecting the ones that have those four bytes as the prefix of their Transaction.input field. The transaction of Listing 1, e.g., corresponds to activity *Customer has a problem*.

Finally, we turn the extracted transactions into a XES document. Every process instance contract corresponds to a trace. Therefore, we use its hash as the trace identifier (id). Every (activity-related) transaction towards that contract maps to an event therein, so its hash represents the id of the event. As an additional attribute we add XES-defined standard

[7] https://etherscan.io/address/0x09890f52cdd5d0743c7d13abe481e705a2706384

Table 3: The activity names, their associated smart contract's function signatures, and the first four bytes of the KECCAK hash of those signatures, for the process in Fig. 1.

Activity name	Function signature	Function selector
Customer has a problem	`Customer_Has_a_Problem()`	`0xefe73dcb`
Get problem description	`Get_problem_description(int32 x)`	`0x92ed10ef`
Ask 1st level support	`Ask_1st_level_support(int32 y)`	`0x82b06df7`
Explain solution	`Explain_solution()`	`0x95c07f19`
Ask 2nd level support	`Ask_2nd_level_support()`	`0x63ad6b81`
Provide feedback for account manager	`Provide_feedback_for_account_manager()`	`0x58a66413`
Ask developer	`Ask_developer()`	`0xecb07b8c`
Provide feedback for 1st level support	`Provide_feedback_for_1st_level_support()`	`0x3b26a0ea`
Provide feedback for 2nd level support	`Provide_feedback_for_2nd_level_support()`	`0x9ec3200a`

```
<event>
  <string key="id" value="i0x656252f3ecee102d981520ca9e0ca0f7048bce99e4f6fead89d358cdbedd6156" />
  <string key="concept:name" value="Customer␣has␣a␣problem" />
  <string key="activity_id" value="e0xefe73dcb" />
  <string key="cost:currency" value="USD"/>
  <date key="time:timestamp" value="2016-03-22T13:44:22.000Z"/>
  <string key="lifecycle:transition" value="complete"/>
  <float key="cost:total" value="0.1754522"/>
  <int key="blockNo" value="1196772"/>
  <string key="from" value="0x1387e74982055e3e1d235aad579350813b329b2b"/>
  <float key="feeEth" value="0.0011393"/>
  <float key="pricePerEth" value="10.96"/>
  <string key="org:resource" value="0x1387e74982055e3e1d235aad579350813b329b2b"/>
</event>
```

Listing 2: XES event corresponding to the transaction of Listing 1.

extensions such as the event time:timestamp, which we approximate by considering the Block.timestamp. We use the Transaction.index to sort events, which belong to the same block. In addition, we include the activity cost:total attribute, on the basis of the consumed gas, plus a correction factor provided by Etherscan for the conversion in US dollars. Furthermore, we include the address of the sender account of the transaction as the org:resource attribute. Listing 2 shows the event extracted from the transaction in Listing 1. The entire generated XES file is available online.[8]

4.2 Analysis of the event log with ProM

We used the generated event log as an input for ProM to test its usage for process mining. A full investigation and analysis of the event log goes beyond the scope of this paper. This preliminary experiment shows the possibilities opened up by our approach, which extracts and trasforms data stored on the blockchain to make it readily available for process mining. Figure 5 illustrates the event log imported in ProM and visualised through the Dotted Chart plug-in. The id of traces is on the y axis and on the x axis we see the timestamp of events. Traces are sorted by the timestamp of their first event. The colour of dots corresponds to the name (activity label) of events. It can be noticed that the process instances were run

[8] https://gitlab.com/MacOS/extracting-event-logs-from-process-data-on-the-blockchain/tree/paper/incident_management_process.xes

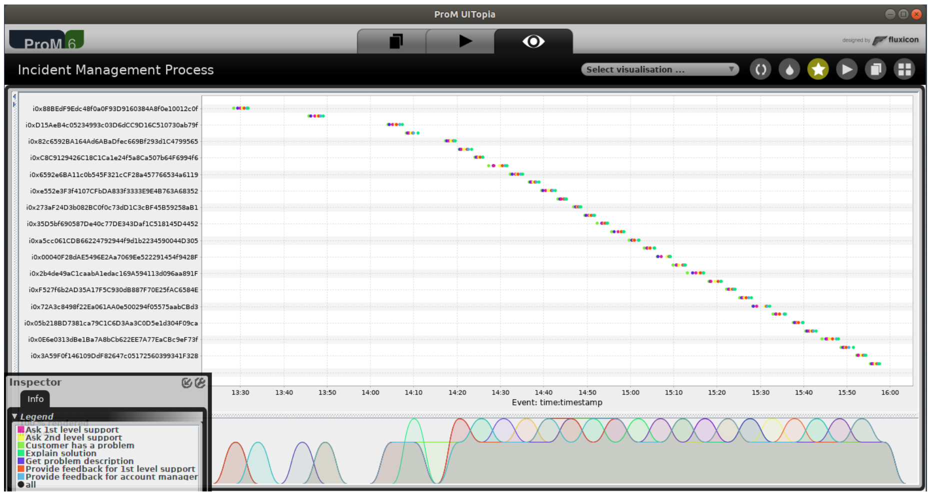

ProM UITopia
ProM 6
Incident Management Process
Select visualisation ...
i0x88BEdF9Edc48f0a0F93D9160384A8f0e10012c0f
i0xD15AeB4c05234993c03D6dCC9D16C510730ab79f
i0x82c6592BA164Ad6ABaDfec669Bf293d1C4799565
i0xC8C9129426C18C1Ca1e24f5a8Ca507b64F6994f6
i0x6592e6BA11c0b545F321cCF28a457766534a6119
i0xe552e3F3f4107CFbDA833f3333E9E4B763A68352
i0x273aF24D3b082BC0f0c73dD1C3cBF45B59258aB1
i0x35D5bf690587De40c77DE343Daf1C518145D4452
i0xa5cc061CDB66224792944f9d1b2234590044D305
i0x00040F28dAE5496E2Aa7069Ee522291454f9428F
i0x2b4de49aC1caabA1edac169A594113d096aa891F
i0xF527f6b2AD35A17F5C930dB887F70E25fAC6584E
i0x72A3c8498f22Ea061AA0e500294f05575aabCBd3
i0x05b218BD7381ca79C1C6D3Aa3C0D5e1d304F09ca
i0x0E6e0313dBe1Ba7A8bCb622EE7A77EaCBc9eF73f
i0x3A59F0f146109DdF82647c05172560399341F328
13:30
13:40
13:50
14:00
14:10
14:20
14:30
14:40
14:50
15:00
15:10
15:20
15:30
15:40
15:50
16:00
Event: time:timestamp
Inspector
Info
Legend
Ask 1st level support
Ask 2nd level support
Customer has a problem
Explain solution
Get problem description
Provide feedback for 1st level support
Provide feedback for account manager
all

Fig. 5: Dotted Chart visualisation of the incident management log in ProM.

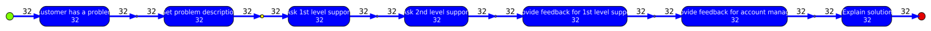
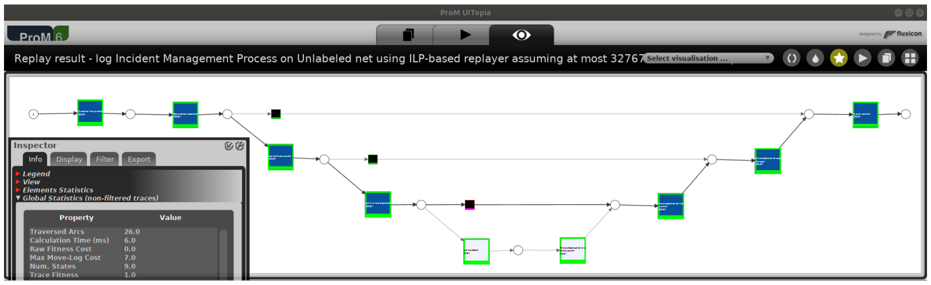


Fig. 6: The incident management process mined via Inductive Visual Miner on ProM.


ProM UITopia
ProM 6
Replay result - log Incident Management Process on Unlabeled net using ILP-based replayer assuming at most 32767
Select visualisation ...
Inspector
Info
Display
Filter
Export
Legend
View
Elements Statistics
Global Statistics (non-filtered traces)

Property	Value
Traversed Arcs	26.0
Calculation Time (ms)	6.0
Raw Fitness Cost	0.0
Max Move-Log Cost	7.0
Num. States	9.0
Trace Fitness	1.0

Fig. 7: Conformance checking of the incident management process on the log.

sequentially at close distance in time. Figure 6 depicts the output of the Inductive Visual Miner discovery plug-in using the standard set-up [21]. Considering the original model in Fig. 1, we can observe that all instances were such that the 2nd-level support resolved the issue as activities *Ask developer* and *Provide feedback for 2nd level support* were never executed. Our observation is confirmed by the application of the conformance checking plug-in of Adriansyah et al. [3] on the log and on a Workflow net that simulates the behaviour of the original process [1], as illustrated in Fig. 7. As it can be noticed in the information panel on the left-hand side, the model is 100% fitting with the recorded traces. Indeed, the border of all activities (transitions, graphically depicted as boxes) are surrounded by a green line. The background colour of transitions indicates whether the corresponding tasks were executed (blue) or not (white). Notice that, as expected, the two activities at the bottom of the diagram (*Ask developer* and *Provide feedback for 2nd level support*) did not occur.

A full-fledged quantitative analysis of the process is beyond the scope of this paper. However, the presented outcome hints at the possibilities that the creation of event logs out of transaction data on the blockchain opens up.

4.3 Discussion and limitations

The described approach shows promising results. However, we acknowledge limitations that are inherently bound to the data analysis we conduct. For process mining, the information found on the blockchain is required to include at least unique identifiers for process instances, timestamps of executions as well as identifiable activity denotations. In our case study, we detect unique identifiers and activity names by considering a smart contract function call as the unit of execution, i.e., the event. However, we understand that smart contract functions may not match process activities. This calls for more advanced techniques to relate smart contract transactions to process task executions [5]. Also, we associate process instances to smart contract life-cycles. This assumption holds when every process run corresponds to one and only one contract. Blockchain BPMSs that adopt other architectural patterns than the factory one could make our assumption not valid any longer, thus requiring more sophisticated reference reconciliation mechanisms, or necessitating transaction payloads to bear a unique identifier for instances. Furthermore, we approximate event timestamps with the block time, although the level of granularity may not be sufficient and fully reliable. Information stemming from off-chain sources should be retrieved from certified sources such as the so-called *oracles* [34]. This opens up new challenges for future work, aimed at mixed on-chain/off-chain information retrieval approaches. In general, our work depends on the data model retrieved from the transactions. In fact, for different cases an adjustment to the corresponding data model may be necessary. In future research endeavours, algorithms from semantic technologies could be leveraged to circumvent manual adjustments [16,8]. Also, we observe that our solution stores the generated event logs off-chain. This could make their synchronisation and update with new process runs harder and potentially hinder run-time monitoring. To circumvent this issue, we envisage solutions that lie at the core blockchain architecture or leverage the support of additional data stores: an extension of blockchain protocol implementations with on-node storage of state and event information [9], or the adoption of hash-based links connecting transaction data with distributed file systems such as InterPlanetary File System (IPFS).[9]

5 Conclusion

In this paper, we described an approach to generate XES-compliant event logs out of process data stored on the blockchain. Our approach provides a blueprint to retrieve process data from the transactions ledger. As a proof-of-concept, we implemented a software prototype applying our approach on a case study, based on a blockchain-enabled process run on the public Ethereum network.

Our approach shows promising preliminary results, on the basis of which we envision a number of future research avenues. From a practical perspective, we are developing further

[9] https://ipfs.io

software artefacts to apply our approach on other blockchain-based BPMSs such as Caterpillar [22]. Furthermore, we aim at investigating how to extract and process information stemming from other Distributed Ledger Technnologies (DLTs) such as Hyperledger Fabric [14].

We argue that the main challenge in mining processes from the blockchain is the mapping between the process-specific data and their model and the concrete representations on the blockchain. Challenges arise if that mapping does not ensure traceability. Future research should thus be devoted to the creation of a modelling language describing how blockchain-based process interactions are mapped onto blockchain data, so as to automate the manual adjustments required by injecting such knowledge in the extraction algorithms themselves. Furthermore, an interesting problem to tackle is the analysis of process data that are partially on-chain and partially off-chain, to include also information sources beyond the reach of blockchains. Novel solutions could build upon the ontology-based data access approach of [8].

Acknowledgements. The authors thank Zhivka Dangarska, Dominik Haas, Jan Mendling, and Ingo Weber for their valuable support and fruitful discussions. The work of Claudio Di Ciccio was partially funded by the Austrian FFG grant 861213 (CitySPIN).

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. van der Aalst, W.M., van Dongen, B.F., Günther, C.W., Rozinat, A., Verbeek, E., Weijters, T.: ProM: The process mining toolkit. In: BPM Demos (2009)
3. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Measuring precision of modeled behavior. Inf. Syst. E-Business Management **13**(1), 37–67 (2015)
4. Bachhofner, S., Kis, I., Di Ciccio, C., Mendling, J.: Towards a multi-parametric visualisation approach for business process analytics. In: Advanced Information Systems Engineering Workshops. pp. 85–91 (2017)
5. Baier, T., Di Ciccio, C., Mendling, J., Weske, M.: Matching events and activities by integrating behavioral aspects and label analysis. Software and System Modeling **17**(2), 573–598 (2018)
6. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The making of KECCAK. Cryptologia **38**(1), 26–60 (2014)
7. Cabanillas, C., Di Ciccio, C., Mendling, J., Baumgrass, A.: Predictive task monitoring for business processes. In: BPM. pp. 424–432 (2014)
8. Calvanese, D., Kalayci, T.E., Montali, M., Tinella, S.: Ontology-based data access for extracting event logs from legacy data: The onprom tool and methodology. In: BIS. pp. 220–236. Springer (2017)
9. Casino, F., Dasaklis, T.K., Patsakis, C.: A systematic literature review of blockchain-based applications: Current status, classification and open issues. Telematics and Informatics **36**, 55–81 (2019)
10. Dannen, C.: Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners. Apress (2017)
11. Del-Río-Ortega, A., Resinas, M., Cabanillas, C., Ruiz-Cortés, A.: On the definition and design-time analysis of process performance indicators. Information Systems **38**(4), 470–490 (2013)
12. 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 (2019)
13. Di Ciccio, C., Cecconi, A., Mendling, J., Felix, D., Haas, D., Lilek, D., Riel, F., Rumpl, A., Uhlig, P.: Blockchain-based traceability of inter-organisational business processes. In: BMSD. pp. 56–68. Springer (2018)

14. Duchmann, F., Koschmider, A.: Validation of smart contracts using process mining. In: ZEUS. CEUR Workshop Proceedings, vol. 2339, pp. 13–16 (2019)
15. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, Second Edition. Springer (2018)
16. Governatori, G., Hoffmann, J., Sadiq, S.W., Weber, I.: Detecting regulatory compliance for business process models through semantic annotations. In: BPM Workshops. pp. 5–17. Springer (2008)
17. Günther, C.W., Verbeek, E.: XES standard definition. Fluxicon Process Laboratories **13**, 14 (2009)
18. Jans, M., Hosseinpour, M.: How active learning and process mining can act as continuous auditing catalyst. Int. J. Accounting Inf. Systems **32**, 44–58 (2019)
19. Kis, I., Bachhofner, S., Di Ciccio, C., Mendling, J.: Towards a data-driven framework for measuring process performance. In: BPMDS. pp. 3–18 (2017)
20. La Rosa, M., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L.: APROMORE: an advanced process model repository. Expert Syst. Appl. **38**(6), 7029–7040 (2011)
21. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Scalable process discovery and conformance checking. Software and System Modeling **17**(2), 599–631 (2018)
22. López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I., Ponomarev, A.: Caterpillar: A business process execution engine on the ethereum blockchain. Software: Practice and Experience (2019)
23. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Balanced multi-perspective checking of process conformance. Computing **98**(4), 407–437 (2016)
24. Mendling, J., Weber, I., van der Aalst, W., vom Brocke, J., Cabanillas, C., Daniel, F., Debois, S., Di Ciccio, C., Dumas, M., Dustdar, S., et al.: Blockchains for business process management-challenges and opportunities. ACM Transactions on Management Information Systems **9**(1), 4:1–4:16 (2018)
25. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
26. Rozinat, A., Van der Aalst, W.M.: Conformance testing: Measuring the fit and appropriateness of event logs and process models. In: BPM. pp. 163–176. Springer (2005)
27. Soffer, P., Hinze, A., Koschmider, A., Ziekow, H., et al.: From event streams to process models and back: Challenges and opportunities. Information Systems **81**, 181–200 (2019)
28. 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)
29. Van Der Aalst, W.: Process mining: Overview and opportunities. ACM Transactions on Management Information Systems **3**(2), 7 (2012)
30. Van Der Aalst, W., Adriansyah, A., De Medeiros, A.K.A., Arcieri, F., Baier, T., Blickle, T., Bose, J.C., Van Den Brand, P., Brandtjen, R., Buijs, J., et al.: Process mining manifesto. In: BPM. pp. 169–194. Springer (2011)
31. 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)
32. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., Weske, M.: Process compliance analysis based on behavioural profiles. Information Systems **36**(7), 1009–1025 (2011)
33. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger (2014)
34. Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A.B., Chen, S.: The blockchain as a software connector. In: WICSA. pp. 182–191 (2016)