Factors Influencing Process Analytics on Distributed Ledgers

Non Peer-reviewed author version

Di Ciccio, Claudio; García-Bañuelos, Luciano; JANS, Mieke; Mendling, Jan; Novotny, Petr & Stage, Ludwig (2019) Factors Influencing Process Analytics on Distributed Ledgers. In: Dumas, Marlon; Hull, Richard; Mendling, Jan; Weber, Ingo (Ed.). Report from Dagstuhl Seminar 18332,p. 86-89.

Table 3 Mapping Capabilities to Features.

Capabilities	Features
Voting	Transaction, Time service, (Anonymization, Notary, Identity Management, Tokens)
Payment	Transactions, Receipts, (Channel, Time service, Tokens)
Asset transfer	Transactions, Tokens, Watermarking, (Channel)
Settlement	Audit trail, Tokens, Notary, Contract
Exchanges	Transactions, Tokens, Assets transfer, Notary
Introductions	Process, Data access, Channel
Referrals	Transactions, Tokens, Identity Management
Reputation	Identity Management, Audit Trails, (Oracles)
Bookkeeping	Audit trails, Receipts, States
Brokering	Identity, Contract, Transactions, State, What-if
Monitoring	Audit trail, Events, Process, Contract (Time Service)
Offering (incl. auctions)	Transaction, Contract, Digital signature, (Time service, pseudonymization)

3. Time service → oracle → Notary;
4. Channel → Identity management → Encryption;
5. Tokens → Transactions.

4.2.5 Conclusion

This summary has taken initial steps towards identifying both the features that can reasonably expect to be supplied by a blockchain platform on the one hand; and the capabilities which applications for that platform may require on the other.

In the future we would like to investigate which features are supported by different blockchain platforms, to guide the decision which platform to choose.

Moreover, as another avenue of research we might look into different solution patterns on how to implement different features.

4.3 Factors Influencing Process Analytics on Distributed Ledgers

Claudio Di Ciccio (Wirtschaftsuniversität Wien, AT), Luciano García-Bañuelos (University of Tartu, EE), Mieke Jans (Hasselt University, BE), Jan Mendling (Wirtschaftsuniversität Wien, AT), Petr Novotny (IBM TJ Watson Research Center – Yorktown Heights, US), Ludwig Stage (Tübingen, DE)

Blockchains trace the sequence of tasks carried out in the course of business process executions by the totally ordered recording of transactions between involved parties, and additionally the logs of events registered by Smart Contracts. This leaves ample room for the ex-post analysis of conducted operations, for analytics, auditing, and mining purposes [40]. However,

it also poses questions on how to design the data storage, keeping into account a.o. the following facts: firstly, the recording of information, or the absence thereof from the stored data, influences the knowledge that can be extracted from ledgers; secondly, the exchange of data in blockchains such as Ethereum is expensive both for what the transactions fees and the write operations from Smart Contracts are concerned [66]; thirdly, saving information as data values carried as transactions payload entails a better traceability, on the one hand, but also unlimited access to the possibly sensible information exchanged, which is detrimental from a privacy viewpoint, on the other hand; finally, multiple blockchains can be adopted that can be either homogeneous or heterogeneous, e.g., a federated network of Ethereum ledgers (e.g. Quorum[1]) for the general inter-organisational, multi-party collaboration, and a federated Hyperledger Fabric for sub-processes involving sub-groups among the participants. On top of this, the introduction of querying languages for data stored as transactional or logging information plays a pivotal role in the introduction of process analytics over blockchains. To that extent, the preliminary contributions provided by the state query languages of Hyperledger Iroha [2], Hyperledger Burrow [3], and the querying of the backing MongoDB database through EOS [4], provide promising results.

In this report, we focus on challenges and requirements for conducting business process analytics on data stored by blockchain-backed process management systems. In particular, we examine the cases in which information is stored fully on-chain or partially off-chain.

4.3.1 On-chain challenges and requirements

In this section we investigate the case in which all the information that is relevant to the process execution is stored on-chain. Discussions on data management and provenance associated with this strategy, including the privacy concerns and the transaction costs, go beyond the scope of this summary. Even under the assumption that the issues related to those topics were appropriately handled, we envisage in the following some crucial aspects to be taken into account for the analysis of this data.

Audit-completeness

Starting with the fully on-chain, single ledger design, the audit-completeness of this data is paramount. Taking inspiration from a requirement set by process mining, if criteria are not provided to uniquely identify the transactions pertaining to a process instance, then linking the evolution of the process becomes a hard manual work at best, thus hampering the process analytics endeavours [6]. In the context of financial auditing, the auditor needs to consider both relevance and reliability of audit evidence. In the context of using blockchain technology, both dimensions might be impacted. The data that is stored on the blockchain ideally contributes to financial reporting assurance (relevant data). Further, providing evidence that data is sufficiently reliable, is often challenging to the auditor [46]. When evaluating this aspect, accuracy and completeness of the data are considered; two aspects that may be impacted positively impacted by blockchain.

[1] https://www.jpmorgan.com/quorum
[2] https://www.hyperledger.org/projects/iroha
[3] https://www.hyperledger.org/category/hyperledger-burrow
[4] https://eos.io/

Eventual consistency

As explained in the CAP theorem [10], distributed systems can enjoy at most two properties out of Consistency (every read receives the most recent write or an error), Availability (every request receives a non-error response), and Partition tolerance (the system continues to operate despite an arbitrary number of messages being dropped or delayed by the network). Reportedly, for instance, Ethereum does not guarantee (strong models of) consistency [4], but only eventual consistency [64]. This signifies that the monitoring of transactions carried out on a local node does not guarantee full reliability. We identify in this context the following *audit patterns of deviation*:

Reordering Transactions, as well as the data they bring, could occur re-ordered in case the local world state in a node gets changed by the substitution of the latest block, or a sub-chain, with a fork that achieved a larger consensus;

Recurring Supposing that a fork lead to a local history rewriting, an already analysed block could be replaced with a new one in which a processed transaction does not occur any longer, yet it recurs in a new mined block thereafter; in such a case, the same information might be included twice if a consistency check is not operated that rearranges the parsed information accordingly;

Missing In the case of forks, transactions that were considered as valid could be excluded by the agreed-upon fork, and then not re-included in case new transactions mined in the new blocks make them invalid; this poses the challenge on whether to discard the retrieved information when the corresponding transaction gets erased from the blockchain.

Abstraction and reverse engineering

In Ethereum, information stored on-chain can occur as event logs emitted from Smart Contracts or as payload to transactions, aside of the internal state of contracts which is however not explicitly written on transaction receipts but has to be recomputed by executing the code locally to the nodes in order to be undisclosed. Event logs and data parameters of the transaction can reveal explicit notifications and context specifications respectively, upon deserialisation[5]. Nevertheless, the way in which logs and exchanged data are engineered is tightly bound to how the the Smart Contracts are encoded. This hampers the ex-post interpretation of those sources of information, let alone their automated analysis. The promised verification and traceability of executed processes ends up being ad-hoc, and demanding manual effort, not so differently from what used to happen when striving to understand the behaviour of legacy systems through their logs [47]. This calls for the introduction of a specification language, possibly using the decorator pattern [22], to enrich the code of methods in Smart Contracts for the sake of self-documentation about state, data, and logging.

4.3.2 Off-chain challenges and requirements

In the remainder, we examine the case in which data are held out of the ledger. Again aside of the considerations on the data management and administration, we portray the envisioned challenges and requirements that involve the merging of operational information retrieved from the blockchain and the affected data from the outside.

[5] https://solidity.readthedocs.io/en/develop/abi-spec.html

Between ledger and the outer data

As the information is split between on-chain and off-chain data, the analysis necessitates of a mechanism to join at least two sources of information, one logging the sequence of actions mediated by methods invoked on smart contracts, the other reporting on the updates on, or retrieval of, data witnessing the conducted tasks. Technologies such as the InterPlanetary File Sysmte (IPFS) provide a mechanism for uniquely linking data chunks spread among peers outside the main blockchain. However, here we refer to those cases in which parts of the data pertaining to the process activities are kept in other systems that can be disconnected from the ledgers, such as external DBMSs, either centralised or federated. Especially in such a case, the association of ontologies to the process specification seems paramount to describe the semantic connections.

Versions of blockchain artifacts

Should the process undergo a redesign phase, the Smart Contracts implementing the old version could be replaced by newer ones. However, whilst the versioning of processes is a feature that is implemented by current Business Process Manaement Systems (BPMSs), the concept of contract replacement is not natively supported by blockchains such as Ethereum. Fully on-chain software architectures such as the one of the blockchain-based BPMS Caterpillar [37] may cater for it, thanks to their implementation of the factory pattern for generating Smart Contracts. However, in partial on-/off-chain scenarios, keeping track of the changes entailed by subsequent versions of the involved artifacts becomes a challenge of higher difficulty, as it involves at once information integration and object matching, together with the semantic version control over software and data updates.

Between ledger and reality

Solutions to connect the digital world of the blockchain with outer reality are crucial to cater for business processes interacting with physical objects, such as in the case of the manufacturing, logistics, or healthcare domains, to mention but a few. For instance, the notion of time is implemented on blockchains such as Ethereum as *block-time*. Such a timestamp is shared among all transactions therein, thus at a coarse-grain level. The aforementioned business processes operate in real time instead. To solve this problem and inject information on real-world information including time, so-called *in-bound oracles* such as Oraclize[6] have been introduced. Oracles operate as a middleware connecting reality with the on-chain information space, and take on the task to return consistent answers to virtually all nodes in the bloakchain that execute the same Smart Contract locally. The out-bound connection seems however more challenging. Owing to the concept of eventual consensus, an operation executed by the run of a Smart Contract may be withdrawn, should a different suffix of the blockchain eventually reach consensus over the local version. From a BPM perspective though, the execution of certain tasks should not be subject to rollback, especially if it leads to permanent changes in the real world. Waiting times could be introduced on purpose between the local transaction and the actual execution of the associated operation in real world, so as to reduce the probability that the task is undone. This is indeed what happens with many purchases paid in Bitcoin. However, this approach might lead to delays that encumber or disrupt the executions of business process in general. Besides, only mitigating the uncertainty of task executions is not sufficient in many cases. An approach that eliminates the risk of operation rollback is thus of high relevance and calls for future research endeavours.

[6] https://github.com/oraclize