Interorganizational Process Execution Beyond Ethereum: Road to a Special Purpose Ecosystem

Christian Sturm[1] and Stefan Jablonski[1]

University of Bayreuth, Bayreuth, Germany
firstname.lastname@uni-bayreuth.de

Abstract. Process automation facilitates a computer-aided distribution of work items to respective participants according to a predefined process model. The automation of interorganizational processes crossing enterprise boundaries requires a consideration of additional issues like the lack of trust among them. Current solutions implement the decentralized control workflow interoperability and rely on the Ethereum protocol for establishing a trusted layer during process execution. In this paper, we specify trust concerns explicitly w.r.t. process perspectives and IT security aspects, dismantle the Ethereum protocol and discuss how the identified components can successfully address the trust issues. An analysis of these components w.r.t. non-functional requirements shows discrepancies between the BPM world and Ethereum and highlights the necessity for tailored designed solutions.

Keywords: Process Execution · Interorganizational Process Management · Decentralized Control · Business Process Management · Blockchain

1 Introduction

Business Process Management provides strategies and tools for companies to organize and perform their operational routines [5]. Workflow models separate application logic from business logic and generate an unobstructed view on the single business processes of a company. Workflow Management Systems (WFMS) automatically interpret the models and care for a compliant distribution of work items to responsible actors. The digital transformation and market globalization pressures companies to rethink and complement their processes to remain competitive [6]: instead of managing internal workflows solely, cross-organizational workflows (e.g. in an interconnected automotive supply chain) must be discovered, modelled and implemented. Contrary to intraorganizational processes, a single central coordination and controlling instance is missing, especially when it comes to the automation and computer-aided execution of these processes.

Global monitoring of the process' progress or trust between participants and a tamper-protected data storage are requirements for feasible interorganizational process management [16]. Former forms of workflow interoperability, i.e. how processes can be assembled into an executable interorganizational process, fail to address all requirements [16]. Decentralized control as novel form promises

to fill this gap by providing a trustworthy common IT infrastructure where every participant remains its autonomy and due to automatic consensus finding, everyone agrees on a common state which is used for passive monitoring and active routing the control-flow based on in-process data values [16].

Ongoing research focuses for instance on the applicability of different blockchain protocols (e.g. Ethereum, Hyperledger, Quorum) in various settings (e.g. supply chain management) [10]. However, we focus on the domain-unspecific abstraction layer of process model-driven workflow automation. So far, related work has followed a top-down approach, where the general purpose blockchain Ethereum was considered applicable and integrated in process execution to establish a trusted layer [12][18]. The maturity and expressiveness of Ethereum justifies the decision in favor of Ethereum in BPM research at first. We argue for a bottom-up approach in future research, where BPM specifies explicit requirements to an architectural layer within the decentralized control context, and tailored solutions are developed and evaluated thereupon. Hence, the strategy in this paper will be to explicitly specify trust issues w.r.t. process perspectives and IT security aspects in Section 3 (functional requirements) as suggested by [14]. Then, we dismantle the Ethereum blockchain into its individual parts in Section 4. In the evaluation in Section 5 we check the suitability of Ethereum components for tackling the identified functional requirements. In the end, we analyze the components also w.r.t. non-functional requirements, recognize limitations and propose guidelines that may help to design special purpose ecosystems.

2 Modeling and Automation of Interorg. Processes

A business process goes through different phases (cf. process life cycle [5]) starting with the modeling using a modeling language, e.g. BPMN (*Business Process Model and Notation*) which enjoys frequent usage in research and industry. BPMN process models include execution semantics so that a WFMS in a company can instantiate and interpret the process model and distribute pending work items to (human) resources accordingly which are responsible for the execution. Hence, the recorded execution history (*event log*) which comprises the actual executed tasks will comply with the process model (cf. Figure 1).

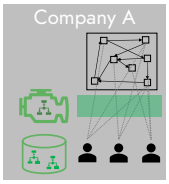

Fig. 1. Process with centralized coordination and single WFMS and multiple local Event Logs
Event Log

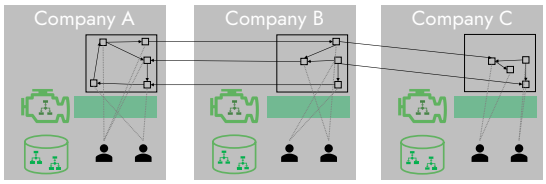

Fig. 2. Interorg. process enactment with multiple

The situation in interorganizational settings is more complicated from a modeling perspective, but also from an implementation point of view due to the lack of a superior controlling authority. Without a central instance which interprets and stores process data globally, interorganizational processes cannot be executed within a WFMS [16]. Instead, BPMN choreographies can specify the required message exchange protocol on a global level, participants orchestrate thereupon their own private local workflows, and notify the respective counterpart in due course according to the choreography [23]. This is implemented using web services (WS-BPEL) and is described in [20] as *loosely coupled workflow interoperability*. However, following this message-flow driven principle, no global progress state is available which hamper automated routing [16] and trust issues on process data may occur [22], e.g. multiple local WFMSs produce multiple local event logs which may be subject to manipulation (cf. Figure 2).

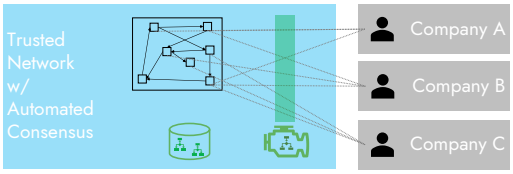

Fig. 3. Automated Consensus ensures Agreement on Data and Process Execution

Decentralized control workflow interoperability solves this issues by providing a trustworthy process-aware network environment which is resistant to malicious manipulation (to some degree). Automatic consensus finding ensures a commonly accepted global state of the process and control-flow driven interorganizational process models can be executed trustworthy (cf. Figure 3). So far, decentralized control was mostly implemented upon the Ethereum protocol [17][18][22]. Instead of bilateral messages as with choreographies and web services, state changes of process diagrams are managed on the blockchain. Because data is kept on-chain, everybody is informed immediately without explicit message exchanges, automatic data-based routing is enabled, and tamper-protection is ensured. Hence, the blockchain assumes henceforth the role of the missing process owner and serves as message broker, work item distributor and ensures conformance.

3 Trust as Functional Requirement

Trust recurs in related work as motivation for the use of a blockchain [22]. Müller et al. provide a structured overview of trust concerns in process execution w.r.t. process model elements and IT security aspects [14]. They have identified trust patterns that addresses the trust issues. We follow a similar approach using these security aspects, but we spotlight trust issues in the context of different

process perspectives [8]. We do not introduce the perspectives here explicitly, but believe that the examples below convey the contents sufficiently to follow the paper. Table 1 summarizes our results and shows good accordance with the work in [14] (cf. the superscripted numbers referring to the trust patterns).

Table 1. Trust Issues w.r.t. Process Perspectives and IT Security Aspects

	Integrity	Availability	Non-repudiation
Functional Perspective	I_f [4.4]	A_f [4.4]	N_f [4.2]
Behavioural Perspective	I_b [4.3]	A_b	N_b
Organizational Perspective	I_{org}		N_{org}
Informational Perspective	I_i [4.1]	A_i	
Operational Perspective	I_{op}		

We refer to Table 1 and describe the trust issues by means of an example from the car service domain. Thereby, we roughly highlight the role of blockchain-based systems during trustworthy process execution.

Functional Perspective A participant can rely on the execution of certain activities, e.g. required checks were executed during car maintenance. From the viewpoint of the executed entity, the execution is not renounceable (N_f), and the on-chain storage also serves as proof for the execution (I_f). Contrary to manual tasks which are executed by human resources, script tasks are executed automatically. Smart contracts on the blockchain ensure a transparent and conform execution (I_f) and the distributed nature of the network leads to a high availability (A_f), i.e. down times are very unlikely in larger networks.

Behavioural Perspective The run-time process flow may be affected by data values or human decisions, e.g. the milage status determines the necessity of various checks or the master craftsman decides on the exchange of parts based on his experience. Data-based routing can be automated in a transparent way and the next tasks are assigned automatically to the respective resource based on the predefined model (I_b). The blockchain also cares for a tamper-protected storage of human decisions or may facilitate voting-based collaborative decisions (N_b). We define the availability of the current progress of the process flow as the global state of the process (A_b).

Organizational Perspective Process models may stipulate a responsibility of an activity to a specific person, e.g. a task must be executed only from a master craftsman. A blockchain-based solution cares then for a globally transparent assignment of the task to the respective resource (I_{org}). If no specific resource is specified a priori, the actor is identified during run-time and the responsible person is logged for traceability purposes. Hence, the person cannot repudiate the responsibility afterwards (N_{org}).

Informational Perspective Data values are produced and consumed from all participants and are subject to possible manipulation. The importance of data requires particular interest in their integrity, i.e. data management that excludes fraudulent modifications (I_i), and availability for data-based routing

(A_i). For instance, the measurement of pollutant emissions produces data which is then decisive for an exchange of affected parts.

Operational Perspective Car workshops can prove the usage of suitable tools (I_{op}). For instance, OBD-II error codes provided by cars to support troubleshooting may include manufacturer controlled diagnostic trouble codes which can only be read properly with the vendor specific reading device. Otherwise errors are diagnosed incorrect.

4 Ethereum Components and their Contribution or Obstruction to feasible Process Execution

In this section, we dismantle the blockchain into its individual parts whilst strongly rely on the work of Tasca et al. [19], where they deconstructed several blockchain protocols to find a common taxonomy. We investigate the components and their responsibilities in resolving the trust issues identified above. Note, that the discussion is independent of any use case but pertains to the abstraction layer of interpreting a process model.

I) Consensus mechanism The consensus mechanism implemented in Ethereum is called Proof-of-Work (PoW). PoW is not responsible for reaching consensus over the data which is stored on the blockchain per se. In the first place, PoW is an algorithm for an automated selection of a participant to be allowed to propose a new set of data which is to be included in the blockchain. PoW is the first algorithm to empower such an election for the first time in a global-scaled network with anonymous (or pseudonymized) participants, whereas naive solutions, e.g. random selection, are affected from sybil attacks: a single user with multiple accounts gets selected more likely [21]. Using PoW, the available computing power is decisive, instead of the number of accounts. PoW is driven by a monetary incentive mechanism. Finding the solution of a cryptographic puzzle is expensive in terms of computing power or electricity respectively, but is awarded with inbuilt cryptocurrency. However, if a participant attempts to include invalid data, the network will repeal the new block and the attacker loses the reward. Hence, PoW ensures network stability by limiting the number of proposed new data and by ensuring that new data can be expected to be valid.

The issue of detection and elimination of faults or intentional manipulation within an untrustworthy network is years old and described for instance as the byzantine generals problem [11] - whose solutions are ascribed to be *byzantine fault tolerant* (BFT). PoW solves the byzantine generals problem in a large scale in the ballpark of thousands of nodes, whereas former fault tolerate algorithms [1][3][15] are afflicted with inferior scalability due to a massive communication overload and are impractical within networks with more than 20 nodes [21].

Lessons Learned BPM currently builds on blockchains incorporating PoW as consensus mechanism and cryptocurrencies and a reward system are necessary in order to run PoW and maintain network stability. In turn, BPM process execution is unnaturally concerned with cryptocurrencies. However, PoW's natural

habitat is a global network with pseudonymized users which does not always comply with the BPM world, where participants are identified and the number of participants in a single process is limited. Classical BFT algorithms also promise to be resistant to a certain fraction of faulty nodes and the issue of bad scalability does not apply in small BPM use cases.

II) Security and Privacy The taxonomy lists *data encryption* as a sub component of Security and Privacy and discusses the selected algorithms (e.g. ECDSA, SHA-2, etc.) mostly. We are interested in the properties of these algorithms and their particular role in the blockchain, especially hashes, private/public-key pairs and digital signatures. In the blockchain, every state change is initiated with a transaction from a user. Transactions are always digitally signed. The sender hashes the transaction and encrypts the hash with his private key to calculate a signature. The triple of transaction, hash and signature is propagated in the network. By applying the decryption function to the signature with the sender's public key and comparing the result with the recalculated hash from the transaction, the network can assume the following statements: the owner of the according private key is the actual sender and the document equals the originally sent document. On the other hand, the sender of a transactions cannot deny being the original sender because the decryption works properly only with his private key.

Lessons Learned In blockchain protocols, an asymmetric key-pair is responsible to sign transactions or to assemble user accounts. In terms of BPM process execution, digital signatures alone may provide enough reliability during process execution, where no full decentralization is required. In practice, participants may then be forced to sign task execution actions (including data) on a centralized or distributed workflow engine. Precise infrastructures must be evaluated and coordinated with respect to the use case.

III) Storage Structure and Counterfeit Protection Data is stored in blocks which serve as container for confirmed transactions. All state changes in a blockchain network are initiated with transactions. These transactions go through different phases: first, they are created locally, before they get propagated in the global network. They are considered valid in this stage as invalid transactions are not propagated and repealed by the network. Valid transactions are stored in the pool of unconfirmed transactions. At some time, a new block with arbitrary unconfirmed transactions is proposed and added to the blockchain and thereby, the transactions get confirmed. To achieve high tamper-protection whilst retaining performance, each new block also includes the hashed information of the predecessor block. Thus, a manipulation of a single bygone transaction is easily detected by checking the latest block only, as the mutations of block headers will propagate. In this regard, the limitation of blocks due to consensus mechanisms like PoW prevents a recalculation of the hash chain, as this is practically impossible hard to solve.

With cryptocurrencies, during the validation of a transaction t_1, the network checks if someone has ever received the monetary units, he is about to spend in past transactions $t_0^0, t_0^1, t_0^2, ...$ The transactions may be included in very ancient blocks wherefore high tamper protection to the very beginning is essential. In the outcome, blockchains suffer from a huge amount of data and transactions are tangled arbitrary distributed over all blocks as second property to mention.

Lessons Learned This storage structure is implemented since the first blockchain which is located in a cryptocurrency context. The design to track all asset transfers to the very beginning and keep them in an immutable ledger is decisive for cryptocurrency-related blockchains as also very old transactions may determine the next valid state. At BPM side, the storage for process data might become obsolete with reaching the end point of a process or when retention periods have expired. Alternative storage concepts may counteract the constantly growth of the database.

The issue of chaotic distribution of all transactions over the blocks is subject to current research, which have to reconstruct the timely-ordered history of task executions for process mining issues [9][13]. Special purpose systems may design storage structures which reflects BPM aware data structures, including process models which vary in time (new versions through enhancement), models that are instantiated multiply, etc. As an example, for a different storage, IoT research introduces a tree-structured data storage to overcome performance issues [4].

IV) Finality A bitcoin transaction is considered totally confirmed after approximately one hour, i.e. six blocks are added afterwards. This high latency, i.e. the time from the insertion of a transaction in a block until the moment of considerable acceptance, is reasoned by the use of practicality-based heuristics. With PoW, confirmation is a progress of increasing probability, not a fixed point in time (non-deterministic *finality*). Non-deterministic or probabilistic finality causes the danger that an appended block could get removed from the current valid chain in case of temporary forks. That is for instance, when multiple valid blocks are propagated simultaneously from different edges of the network. Such abnormalities are dissolved following the longest chain rule. As a result, validated and verified transactions may get reverted when they are not included in the longest chain. Classical BFT algorithms are not affected as they inherit consensus finality [21].

Lessons Learned The solution in current blockchains is to simply recreate and re-propagate removed transactions. What is easy in the cryptocurrency world, is much more complex regarding process execution. With non-deterministic finality, BPM must answer questions according to the task life cycle, for instance when should the next task be claimed or executed? at time when the respective transaction occurs in the mining pool, when it gets included in a block, or when enough succeeding blocks are mined? Another question concerns the (dirty) redo of a task. It cannot be presumed, that every task can be redone. Hence, deterministic finality should be a design principle in a BPM related ecosystem.

V) Extensibility The level of extensibility determines how flexible the rules for considering a transaction valid can be modified [19]. In bitcoin for instance, a transaction to be valid must reference to incoming monetary units of the respective account for each outgoing monetary unit. The flexibility culminates in turing-complete execution environments, like the Ethereum Virtual Machine, where arbitrary conditions can be defined. This serves many application areas and is surely responsible for Ethereum's success story. As being turing-complete, the systems consequently suffer from denial-of-service (DoS) attacks. To confirm a transaction, a node must execute the assigned code. For turing-complete environments there is no algorithm which can check the termination for arbitrary programs (halting problem). To counteract, Ethereum introduced *gas* which is related to the inbuilt cryptocurrency *Ether* and thus an equivalent to real money. Every transaction is associated with an amount of gas which represents the maximum computing effort which is available to confirm the transaction. Hence, participants pay for the confirmation which eliminates the danger of DoS attacks.

Lessons Learned For process execution itself, turing-completeness is not a viable requirement. A BPM execution environment must only support the execution semantics regarding a modelling language. Consequently, DoS attacks could be eliminated by design, instead of synthetically prevent them with gas and introducing a cryptocurrency like in Ethereum.

VI) Cryptocurrency, Charging and Rewarding System Cryptocurrencies are the central use case for the first blockchain (bitcoin). It was promoted as digital payment instrument which dispense with third party providers like banking institutions but is under full control of the network. However, the cryptocurrency topic is addressed in various ways in different blockchain protocols. See [19] for further details. We have covered a subset of use cases in the above discussion. Cryptocurrency drives important pillars, e.g. the rewarding system as incentive for mining new blocks (network stability through PoW) or the charging system by paying for transactions to prevent DoS attacks (gas).

Lessons Learned Cryptocurrencies are foreign to process execution in the first place, but other components of the blockchain rely on cryptocurrencies. Hence, to eliminate this component, alternative implementations for other components must be included. On the other hand, research has already shown, that cryptocurrencies can also automate payment transactions during process execution [22].

5 Evaluation

The initial research goal was to secure process execution and settle disputes automatically. Thereby, research has established trust as an umbrella term. We have unraveled trust in Section 3 and dismantled Ethereum in its components in Section 4. In this section, we synthesize both sides and evaluate which components are responsible for a specific trust dimension. The result is further on investigated from a non-functional point of view.

5.1 Blockchain Components satisfy Functional Requirements

Section 3 defines trust in terms of IT security aspects and process perspectives. Table 2 aggregates over the later and shows the responsibilities of components from the blockchain system for certain IT security aspects during process execution. The integrity aspect was split into run-time conformity (w.r.t. the process model) and the no-modifications aspect of event logs as suggested in [2].

Table 2. Responsibilities of Blockchain Components w.r.t. Security Aspects

	Conformity	No Modifications	Avail. of Data	Non-rep.
Cons.	Validation rules	Longest Chain Rule	×	×
Sec.	Asym. Crypt.	Hashing	×	Digit. Sig.
Ext.	Process Semantics	×	Script tasks	×
Stor.	×	Chaining of Blocks	×	×
Fin.	×	×	Agreement	×
Crypt.	×	×	×	×

Conformity Each action of users with the blockchain (transactions) must match the validation rules to be considered during consensus finding (Cons.). Hence, non-compliant actions will not affect the process status. The validation rules are defined in terms of extensibility (Ext.) and regarding the organizational perspective, resources can be identified with their private key (Sec.).

No Modifications It is important that data cannot be modified, once agreed upon. The blockchain is ascribed to be tamper-resistant, because modifications are detected easily by having a representation hash of all data in one block (Sec.) and chaining these hash values through the blocks. Even small modifications would propagate to the latest block (Stor.). It is theoretically possible to convince the network that the malicious modification should be accepted: an attacker must then recalculate the whole hash chain from the block containing the respected transaction or data until the latest block, so that the modification is still in the longest chain, but the difficulty of PoW to find valid blocks prevents this (Cons.).

Availability of Data Tasca et al. defines finality when information intended to be stored in a blockchain can be safely considered perpetually stored. This is, when every participant in the network agrees on the same state of the blockchain and thus on the same state of the process' progress (Fin.). At this point in time, all data and information are unlikely to get reverted and can be considered available. The availability of single (automated script) tasks is given, because every node in the network is able to execute the assigned code (Ext.)

Non-repudiation Every actor in the network must add a digital signature to each transaction. This signature is associated with the actor's private key and hence, he cannot deny being the initiator of the transaction, because the signature can be verified with his corresponding public key solely (Sec.).

5.2 Analyzing Non-functional Concerns of Ethereum Components

Some components cannot directly be ascribed to tackle a certain security aspect, e.g. cryptocurrency. However, cryptocurrencies are an important mainstay during Ethereum process execution because they are the engine behind dependent components which are necessary to reach the addressed goals. As stated, the functional requirements are satisfied, however going with Ethereum means that we have to take a loss in terms of non-functional observations which are subject to this section. Table 3 gives an overview on which Ethereum components influence selected non-functional characteristics.

Table 3. How do Components influence Non-functional Characteristics

	Performance	Costs	Avail. of System	Scalability
Cons.	Block Period	×	Rationed Data Proposals	PoW
Sec.	×	×	×	×
Ext.	×	Gas	Gas	×
Stor.	Tangled Txs	Storage costs	×	×
Fin.	Final Validation	×	×	×
Crypt.	×	Tx fees	Incentive	×

Performance The consensus mechanism in Ethereum is currently configured so that a new leader node is selected, and a new block is proposed every 10-20 seconds (Cons.). Due to probabilistic finality, transactions are considered perpetually stored after 37 confirmations in succeeding blocks [7] which equals to approx. 6-12 minutes and causes a severe performance issue (Fin.). An idle time of up to 12 minutes after each task to have persistent data is not acceptable from our point of view. A secondary issue are tangled transactions which may inhibit performance during analysis (Stor.).

Costs Having a turing-complete scripting language in the extensibility component, the Ethereum protocol introduces gas, a measure with financial countervalue, to avoid DoS attacks (Ext.). Additionally, actors have to pay a small fee for each transaction (also with financial countervalue in terms of cryptocurrency) which will be transferred to the winner of the PoW puzzle as allowance (Crypt.). Also storing data on chain carry costs (Stor.).

Availability of System The system is of global-scaled nature and the availability time and network stability is driven by the PoW consensus mechanism in terms of the rationed data proposals every 10-20 seconds (Cons.) and the incentive mechanism for compliant behaviour (Crypt.). Speaking from availability, DoS attacks are prevented with gas (Ext.).

Scalability Scalability is a broad term and may affect in the blockchain context the number of nodes, transactions, users, etc. [19]. In BPM context, we define this as the number of participants in consensus finding which is not a limiting factor with PoW (Cons.).

5.3 Guidelines for a Bottom-up System Design

Until now, we have shown how blockchain components address the identified trust issues and that blockchain components come along with severe restrictions w.r.t. non-functional aspects of process execution. Consequently, the Ethereum blockchain should not be considered as silver bullet. For a special purpose system design, the following questions may help to specify guidelines which we plan to verify in future research in a more structured way.

In a global network with full decentralization and many participants, a highly scalable consensus mechanism like PoW must be included, which inevitable comes along with cryptocurrencies and probabilistic finality. Ethereum can also be configured as permissioned network with Proof-of-Authority consensus finding. Then again, trusted nodes must be included to verify data which limits the level of decentralization. In small-scaled networks up to 20 nodes, BFT algorithms can be considered for consensus finding. In minimal bilateral or trilateral relations, decentralization fades into the background and *trust* during interorganizational process execution may be established with sole digital signatures.

A sophisticated automation of tasks (script tasks) requires a powerful computing environment which must be protected from DoS attacks by introducing a cryptocurrency. However, speaking of a system which is mainly used for global routing and task distribution, the expressiveness can be limited to interpret process language and DoS-attacks and hence cryptocurrencies are eliminated by design.

If no long-term data storage and protection is required as it is the case with cryptocurrencies, alternative storage structures can be developed instead of using the continual growing blockchain.

6 Conclusion

Interorganizational process management strives for a trustworthy environment during process execution. Decentralized control acts in a P2P-network with automated consensus finding to address these trust issues. First implementations rely on the complex Ethereum protocol. We have shown, that the building blocks in the ecosystem may influence process execution positively (e.g. decentralization, automated consensus finding in the presence of malicious nodes, tamper-protected data storage) but also in a negative respect for certain use cases (cryptocurrency, overpowered scripting language, latency). The evaluation of trust issues as functional requirements w.r.t. blockchain components resulted in discrepancies between the Ethereum world and the BPM world, especially the further analysis on non-functional aspects. Hence, the necessity of a special purpose solution is demonstrated.

Future BPM research must concentrate on a detailed specification of trust related requirements and tailored solutions must be selected in favour of the general purpose Ethereum blockchain, which despite solving the initial research question comes with not negligible reservations w.r.t. non-functional concerns.

References

1. Bessani, A., Sousa, J., Alchieri, E.: State machine replication for the masses with bft-smart (2014)
2. Biskup, J.: Security in Computing Systems (2009)
3. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: Proc. of the Third Symposium on Operating Systems Design and Implementation. OSDI '99 (1999)
4. Dorri, A., Jurdak, R.: Tree-chain: A fast lightweight consensus algorithm for iot applications. PREPRINT (2020)
5. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, Second Edition. Springer (2018)
6. Fridgen, G., Radszuwill, S., Urbach, N., Utz, L.: Cross-organizational workflow management using blockchain technology (2018)
7. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proc. of the 2016 ACM SIGSAC Conference on Computer and Communications Security
8. Jablonski, S., Bussler, C.: Workflow Management: Modeling Concepts, Architecture, and Implementation (1996)
9. Klinkmüller, C., et al: Mining blockchain processes. In: BPM: Blockchain and Central and Eastern Europe Forum (2019)
10. Kumar, A., Liu, R., Shan, Z.: Is blockchain a silver bullet for supply chain management? Decision Sciences **51** (2020)
11. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Transactions on Programming Languages and Systems (1982)
12. López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I., Ponomarev, A.: Caterpillar: A business process execution engine (...). Softw. Pract. Exp. **49** (2019)
13. Mühlberger, R., Bachhofner, S., Di Ciccio, C., García-Bañuelos, L., López-Pintado, O.: Extracting event logs for process mining... In: BPM Workshops (2019)
14. Müller, M., Ostern, N., Rosemann, M.: Silver bullet for all trust issues? blockchain-based trust patterns for collaborative business processes. In: BPM: Blockchain and Robotic Process Automation Forum (2020)
15. Schneider, F.B.: Implementing fault-tolerant services using the state machine approach: A tutorial. ACM Comput. Surv. **22** (Dec 1990)
16. Sturm, C., Scalanczi, J., Jablonski, S., Schönig, S.: Decentralized control: A novel form of interorganizational workflow interoperability. In: The Practice of Enterprise Modeling - 13th IFIP Working Conference, PoEM 2020, Proceedings (IN PRESS)
17. Sturm, C., Scalanczi, J., Schönig, S., Jablonski, S.: A blockchain-based and resource-aware process execution engine. FGCS Journal **100** (2019)
18. Sturm, C., Szalanczi, J., Schönig, S., Jablonski, S.: A lean architecture for blockchain based decentralized process execution. In: BPM Workshops
19. Tasca, P., Tessone, C.J.: A taxonomy of blockchain technologies: Principles of identification and classification. Ledger **4** (Feb 2019)
20. van der Aalst, Wil: Process-oriented architectures for electronic commerce and interorganizational workflow. Information Systems **24** (1999)
21. Vukolić, M.: The quest for scalable blockchain fabric: Pow vs. bft replication (2016)
22. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: Business Process Management - 14th International Conference, Proceedings (2016)
23. Weske, M.: BPM - Concepts, Languages, Architectures. Springer (2019)