Flexible Integration of Blockchain with Business Process Automation: A Federated Architecture

Michael Adams¹, Suriadi Suriadi¹, Akhil Kumar², and Arthur H. M. ter Hofstede¹

¹ Queensland University of Technology, Brisbane, Australia
² Smeal College of Business, Penn State University, University Park, USA

Abstract. Blockchain technology enables various business transactions to be performed in an immutable and transparent manner. Within the business process management community, blockchain technology has been positioned as a way to better support the execution of inter-organisational business processes, where the entities involved may not completely trust each other. However, the architectures proposed thus far in the literature for blockchain-enabled business process management can be described as “heavy-weight”, since they promote the blockchain platform as the monolithic focal point of all business logic and process operations. We propose an alternative: a federated and flexible architecture that leverages the capabilities of blockchain, but without overloading the functionalities of the blockchain platform with those already extant in Business Process Management Systems (BPMSs). We illustrate its benefits, and demonstrate its feasibility, through the implementation of a prototype.

Keywords: blockchain; process flexibility; business process automation; business process management systems.

1 Introduction

A blockchain is a tamper-proof, replicated and distributed ledger [10] to which multiple parties can append transactional records in such a way that modification is prevented, in a technically-enforceable manner. Blockchain technology effectively guarantees that transactions, once recorded, become immutable [17], facilitating the execution of transactions across multiple, potentially untrusted parties without the need for a trusted intermediary. Naturally, blockchain opens up new opportunities to support the execution of cross-organisational business processes (i.e. those processes that necessitate interactions involving multiple discrete players) typically seen in many domains, such as supply chain management and manufacturing.

In recent years, the Business Process Management (BPM) community has investigated ways to exploit blockchain for secure, cross-organisational process execution (see [6–8, 19] for some initial approaches). In this paper, we specifically focus on the alternative architectural designs that integrate blockchain
technologies with business process management systems (BPMS) to support process executions involving multiple, independent parties. We call such a system a blockchain-integrated BPMS.

A prominent architecture proposed for blockchain-integrated BPMS transforms a business process, expressed as one or more process models, into smart contracts (programmable transactions) that are then executed entirely upon a blockchain platform [6, 19]. That is, all business rules, branching logic, instance data, resource allocation, access authorisations and process state management is deployed to and handled by the blockchain platform. Thus, the focal point of this architecture resides in the blockchain and the different parties involved in the business process must interact directly with this blockchain, both during process design time and runtime executions. We shall refer to such an architecture as blockchain-centric.

While blockchain-centric architectures may be appealing for some business process applications, and under certain threat assumptions and/or risk scenarios, it is not a universal solution. It is a heavy-weight architecture, with a rigidity that may not be necessary, or even desirable, in many other business process applications, for example where interactions between multiple parties are loosely-coupled and/or may involve asynchronous-type interactions. A heavy-weight architecture also overloads a blockchain system with a host of supporting compilers, components and mechanisms required to wholly accommodate business process design and execution within a distributed ledger. In effect, this tight integration necessitates a duplication of the capabilities that already exist within core execution engines of BPMSs.

Hence, we propose an alternative federated architecture, that is more decentralised, cooperative, and flexible, simpler to realise and better suited to meet the needs of a wide variety of cross-organisational settings. The proposed architecture allows component parts to each perform their fit-for-purpose capabilities in a federated whole, rather than overloading components with functionalities that are better performed by others. This architecture is flexible in that it is not tied to any particular type of blockchain platform or BPMS. It supports the minimisation of append operations to a blockchain, which are known to be resource-intensive [18, 21], and does not require the creation and propagation of multiple smart contracts per execution instance.

This paper is structured as follows: a background discussion and related work are presented in Section 2, Section 3 establishes the need for a federated blockchain-integrated BPMS, and Section 4 describes the proposed architecture, while Section 5 illustrates its implementation. A comparative discussion of the federated and blockchain-centric approaches is presented in Section 6, followed by the conclusion of the paper.

2 Background and Related Work

The key advantages of blockchain, besides immutability, include: visibility (all authorised participants can view the transactions); validation (transactions are
endorsed by peers through a designated consensus mechanism prior to being written to the chain); and resilience (a replicated ledger means there is no single point-of-failure).

A blockchain system can be permissioned (exercise membership control) or permissionless (publicly accessible). For example, Ethereum\(^1\) \[^3\] is (by default) a permissionless blockchain platform, where any peer can join to read or submit transactions at any time. Moreover, there is no central entity to manage membership, although private and permissioned blockchains can also be configured. Permissioned blockchain systems are designed to better address concerns around transaction security, privacy and scalability \[^1\]. Hyperledger Fabric\(^2\) \[^4\] is an example of a permissioned blockchain framework. Another key aspect of blockchain technology is the provision of so-called smart contracts \[^5,15\], i.e. executable scripts that reside on the blockchain and automate the steps and rules corresponding to the business logic of the bespoke transactional operations.

In recent research efforts towards integrating blockchain technology with BPM \[^6,10,19\], the authors propose an architecture that tightly integrates business process execution with blockchain by encapsulating the entire business process logic into smart contracts. In this approach, a translator component takes a process specification as input and generates a set of corresponding smart contracts per process instance. In addition, a choreography monitor uses smart contracts to control a collaborative business process. A prototype has been developed for the Ethereum platform \[^19\].

Architectural design issues of blockchain based systems with an eye towards quality and performance attributes are addressed in \[^20\] in the form of a taxonomy and flowchart. Other performance issues that have been addressed are availability \[^18\] and latency \[^21\]. Methods for optimising execution of business processes on an Ethereum blockchain by improving data structures and runtime components are discussed in \[^6\] and demonstrated in a prototype called Caterpillar \[^8\]. Approaches for implementing collaborative, data-aware business processes on blockchain using the business artifact paradigm are discussed in \[^2,7\], focussing on a new business collaboration language.

Sturm et al. \[^14\] develop a generic approach to control-flow management within the blockchain by having one contract that handles choice and parallel structures. However, the control-flow capabilities are limited and data management is not discussed. There are also a plethora of approaches to inter-organisational process management that use platforms and environments other than blockchain, for example \[^9,11,13\].

All these related approaches have helped to locate our work in context. However, our approach is different in that we believe that the essential functionality of a BPM system should not be migrated to the blockchain. Instead, we explore a lean approach (along the lines of \[^14\]) wherein the BPM system can interface with the blockchain as a repository of reliable data and for executing key contractual terms through smart contracts.

\(^1\) https://www.ethereum.org/  
\(^2\) https://www.hyperledger.org/projects/fabric
3 Towards a Federated Blockchain-integrated BPMS

Consider the pharmaceutical use case scenario shown in Figure 1. In this cross-organisational process, a **Pharmacy** places an order for medical supplies with its **Distributor**, who in turn requests the production of the pharmaceuticals by the **Manufacturer**. Once the pharmaceuticals are manufactured, they are delivered to the Distributor who then sends them to the Pharmacy.

When this process is executed, there is a potential for conflict across different parties. For example, if the Distributor fails to deliver the ordered pharmaceuticals on time, the Distributor may blame the Manufacturer for being late with production, or the Distributor may dispute the date and time when it received the original order. Therefore, the use of blockchain in recording the process transactions can be beneficial. Moreover, each organisation can exercise full control over their own private business process, and share information of only selected activities that involve cross-organisational interactions, as shown in Figure 1.

There are many desirable features of this approach. Firstly, the parties in the process do not need to agree on a common inter-organisational process. They may even be on different blockchain platforms so long as they are compatible. Secondly, the lower transparency requirement will increase the willingness of the
parties to cooperate with each other. Thirdly, there is more scalability in such an arrangement since in general a pharmacy will deal with multiple distributors, and a distributor, in turn, with multiple manufacturers. Thus, this use case calls for a more flexible, decentralised, loosely-coupled and distributed approach based on platform heterogeneity, for both two-party and multi-party interactions, which minimises the need for interactions with the blockchain platform.

Towards a Federated Approach We propose a federated, blockchain integrated BPMS architecture to address the issues identified above. Such an architecture should provide the following properties:

- **Separation of Concerns:** A clear separation of capabilities should be maintained between business logic operations and distributed transactional execution records, with the aim of minimising the performance hit on blockchain operations and maximising the fit-for-purpose capabilities of the BPMS and blockchain platforms.

- **Platform Heterogeneity:** The architecture should allow the use of more than one compatible blockchain platform within and across a composite set of interacting process instances.

- **Compartmentalisation of Interactions:** A requirement that all interactions between any two participating parties need to be transparent to all parties involved should not be imposed. A blockchain-centric architecture may perhaps support this through the use of, for example, separate permissioned channels, but this should not be seen as a necessary realisation, and it still imposes the requirement that they share the same blockchain platform.

- **Single-party Interaction:** The architecture should not assume that all interactions between a business process and a blockchain involve multi-party communication. Hence, it should support simple single-party interaction between an organisation’s business process and its corresponding blockchain.

4 Conceptual Architecture

In our federated approach, each organisation hosts a discrete BPMS that encapsulates a service or middleware component through which it will delegate designated tasks, designed to perform a required inter-organisational activity, within a process execution instance. The service will then interact with a properly configured blockchain network.

Each participating service in an inter-organisational process is granted authorisation to a discrete permissioned channel (or other authenticating, secure pipeline) on a blockchain network. A channel is a private overlay that partitions a blockchain network to provide data isolation and confidentiality [1]. Whenever a new block is written, an event notification is generated by the blockchain platform and then relayed to the BPMS through the service. The service will by default listen for events as they occur, but it may also be configured to periodically request the event history from past blocks, to accommodate those
deployments where connection to the blockchain network is not always available. The service will take one of three actions for each received event notification, depending on how the service has been configured for each event: (1) release a task that has been waiting for the event to occur; (2) launch a new process instance, using the event as a trigger; or (3) ignore the event.

Hence, the only information exchanged between organisations is that required for work to be handed over and performed within each organisation (e.g. purchase order, invoice, contract, application). The state of a process instance can be inferred from the history of data associated with it on the blockchain, for example an order has been placed, a shipment was sent, a payment was made, etc. This eliminates the need for sharing additional information about exact process state on the blockchain, or any process definitions, business logic and rules, organisational data, or resource allocations that should remain private to their respective organisations.

A transaction (such as placing a purchase order) submitted by one organisation to the blockchain will, within a short period, be written to a block on the blockchain after it is validated by other peer nodes on the network using a validation algorithm, and ordered along with other transactions into a block structure. The creation of a new block will trigger an event notification which may be used by another organisation to complete a task in one of its own processes or to commence a new process instance (see Section 5 for more details).

An internal architecture of the proposed approach is given in Figure 2. The BPMS of an organisation will delegate the execution of certain tasks to the blockchain service (middleware component) using the appropriate API along with the requisite data. The subcomponents of the middleware are:

- **Smart Contract Invoker**: Performs smart contract calls on the blockchain to either query the current instance data that has been written to the chain,
or requests the creation of a new transaction to store data to be shared with another organisation.

- **Event Listener**: Listens and responds to events generated by the blockchain network each time a new transaction is created. An event may trigger the completion of a waiting task, or the launch of a new process instance via a call to the BPM engine’s API.

- **Task Cache**: Stores tasks that are waiting for some event to occur on the blockchain, that is some specific data to be made available from another organisation (e.g. order received, invoice produced, etc). When the designated event occurs, that task can be further processed and/or completed, allowing its parent process instance to continue.

- **Authority Certificate Store**: Stores the private and public keys authorising the service to access the channel to read from and submit to the ledger on behalf of its owner organisation. Each call of a smart contract must be accompanied by the appropriate certificates.

## 5 Implementation

A prototype service that implements the conceptual architecture described in Section 4 has been realised in the YAWL business process management environment [16]. YAWL was selected because it is robust, fully open-source, and offers a service-oriented architecture, allowing an interactive blockchain service to be implemented independent of existing components. However, the generic federated architecture is not limited to the YAWL environment, but rather is applicable to any BPMS that supports the addition of service-oriented or middleware components for interacting with external networks and applications. Importantly, absolutely no changes were required to be made to the YAWL environment itself to enable support for communication and interaction with a blockchain network. The YAWL Blockchain Service, and its source code, can be freely downloaded from the YAWL repository³.

For this prototype implementation, a Hyperledger Fabric blockchain network was chosen, because it is open-source, can be deployed freely, does not require crypto-currency payments for its operations and supports a permissioned network natively. Again, the architecture is not limited by this choice; other blockchain platforms may be used.

The Blockchain Service has been developed as a YAWL custom service and so may have tasks assigned to it at design time via the process editor. At runtime, the engine delegates all such assigned tasks to the service for action, passing task input data and metadata to it via a specific process engine API. Communication between the YAWL Blockchain Service and the Hyperledger Fabric network is handled via the Java software development kit (SDK) for Hyperledger⁴. Each organisation maintains its own discrete YAWL environment and Blockchain Service.

---

³ [https://github.com/yawlfoundation/yawl](https://github.com/yawlfoundation/yawl)

⁴ [https://github.com/hyperledger/fabric-sdk-java](https://github.com/hyperledger/fabric-sdk-java)
5.1 Event Handling

The architecture leverages the event generating capabilities of the blockchain platform to provide change-of-state announcements in the end-to-end process, in particular notifying parties in cross-organisational processes of actions that have been taken by others. Events of interest can be used to release a task that has been waiting for an action to occur, or can signify the triggering of a new process instance within an organisation.

For a task that has been designated to wait for an action, a dedicated data structure, which specifies the event to wait for and the values that uniquely identify the event as related to the current end-to-end process instance, is included as an input parameter of the task. On being delegated the task at runtime, the Blockchain Service stores details of the task, and the specifics of the \textit{WAIT} data values, and compares each incoming event with those parameters. When a match occurs, the task is updated with the values attached to the event, and released, i.e. returned to the core BPMS engine, allowing the process instance to continue.

The other type of event of interest to the Blockchain Service signifies the triggering of a new case instance. For example, if a pharmacy raises a purchase order and submits it to a blockchain, the event produced by the blockchain when the order transaction is written to a block can be captured by the Blockchain Service of a pharmaceuticals distributor and used to trigger the creation of a new process to fulfil that order (see Section 5.2 below for more details). Events can be defined as process triggering events via a dynamically loaded configuration file, or via an input data variable for a task, or by using an administration tool.

5.2 Illustrative Example

An example execution of typical interactions among the three organisations in the supply chain scenario of Section 3 is illustrated in Figure 3. The processes have been somewhat simplified for clarity in the discussion below, and are depicted in the YAWL language.

There are three interacting organisations: a \textit{Pharmacy} that places orders for the supply of pharmaceuticals, a \textit{Distributor} that fulfils those orders, and a \textit{Manufacturer} that fabricates and supplies pharmaceuticals for distribution. The Pharmacy interacts only with the Distributor, similarly the Manufacturer interacts only with the Distributor, and consequently the Distributor interacts with both. To ensure data isolation and confidentiality, two channels are created, one \textit{Pharmacy} $\leftrightarrow$ \textit{Distributor} (called \textit{chPharmDist}), the other \textit{Manufacturer} $\leftrightarrow$ \textit{Distributor} (\textit{chManuDist}). Importantly, all internal processes remains private to each organisation, only the transactional data necessary to collaborate with another organisation is shared via the blockchain.

While this scenario concerns a specific pharmacy-distributor-manufacturer, more generally a distributor would deal with a number of different pharmacies and manufacturers, and vice versa, all of which would potentially participate as peers within the blockchain platform and may play a role in the validation
consensus that occurs when a transaction is submitted to the chain. Further, it is of course also possible to have a single channel for all three parties if desired.

To illustrate a complete sequence of interactions, with reference to Figure 3 and the numeric labelling within it:

1. A composite process instance begins with the pharmacy process, when a new order is generated and then sent, i.e. submitted and subsequently written to a new block on the chain via a task delegated to its YAWL Blockchain Service.
2. Since the permissioned channel $\text{chPharmDist}$ is shared by the Pharmacy and the Distributor, the Distributor's Blockchain Service detects the new $\text{BlockEvent}$ and interprets it as a trigger to launch a new instance of its internal ‘supply’ process. The transaction data sent with the event (i.e. the purchase order) is used as the originating data for the new instance.
3. The Distributor adds the order to a batch, then at a designated time submits the batch order to the Manufacturer via submission to the blockchain via the shared $\text{chManuDist}$ channel shared by those two organisations.
4. The Manufacturer’s service receives the write $\text{BlockEvent}$, which triggers a new instance of its own ‘manufacture’ process, using the transaction data in the event (i.e. the batch order) as originating data.
5. Once the Manufacturer ships the order, the process archives the order details on the blockchain.
6. This $\text{BlockEvent}$ triggers the release of the waiting Receive and Verify task in the Distributor’s process, allowing that process to continue.
7. Later, an invoice is produced by the Distributor and submitted to the blockchain via the $\text{chPharmDist}$ channel.
8. The subsequent writing of the invoice to the chain causes a $\text{BlockEvent}$ that triggers the release of the waiting Receive Invoice task in the Pharmacy’s process.
9. Eventually, the Pharmacy pays the invoice by submitting the payment transfer details to the chain.
10. The payment causes a BlockEvent that triggers the release of the waiting Receive Payment task in the Distributor’s process.

Significantly, this example illustrates that secure inter-organisational process automation can be achieved using a federated architecture, and that the approach affords several concrete advantages when compared to the more heavyweight, blockchain-centric architectures:

- Efforts to combine the three processes into one overarching, monolithic, end-to-end process model are no longer required, negating the need for a great deal of collaboration between all parties, and the translation of the result into a set of factory smart contracts. The architecture also avoids the need for the creation, verification and storage of a new set of smart contracts for every instance of the inter-organisational process.
- Because all business logic, branching rules, resources allocations, etc. are handled by the BPMS, the smart contracts here are not overloaded with procedural code, resulting in much simpler, faster to process transactions. In this example, the smart contracts define data structures for order, invoice and payment, and a trivial invoke function that either submits a transaction or performs a query over existing blocks. The data structures are used to (de)serialise JSON strings passed to/from the BPMS into block data.
- There is no need to ‘centralise’ the process on the blockchain. Each organisation retains autonomy of its own processes, and the foci of operations are retained within the processes of each organisation’s BPMS.
- There is no requirement for the creation and maintenance of the “intricate set of components” [19], prerequisite to the heavyweight architecture. Only a standard BPMS environment, a simple middleware service and vanilla blockchain network are needed.
- Unlike many blockchain-centric architectures, there is no requirement for a central ‘mediator’ process to choreograph the interactions between each organisation’s processes.
- There are no limitations placed on the types of process patterns supported. Any pattern supported by the process language used by the process execution environment (i.e. the BPMS) can be used in this approach, including those more complex patterns that are difficult, if not impossible, to transform into a smart contract, since all process executions are contained within the BPMS, rather than on the blockchain.
- An unimpeachable audit trail is stored on the blockchain(s) and can be extrapolated for all inter-process activity instances between organisation pairs.

6 Discussion and Conclusion

Many blockchain-centric approaches use a blockchain monolithically as an entire execution platform for business processes. Thus, a potentially large volume of data, including process definitions in the form of smart contracts, business rule definitions, datasets representing the work of a process instance, as well as its
constantly updating state information, is written to, read from, and executed on the blockchain. Depending on the smart contracts and business rules executed, such data could contain potentially-confidential internal data of an organisation, thus inadvertently and unnecessarily exposing private data to external parties.

As per our case example (Figure 3), it will require 20 large, custom contracts to be created (one for each task) in such blockchain-centric architectures versus 7 short, generic contracts that merely write important transactions to the blockchain in our approach. Additionally, each custom contract will require considerable effort for verification, and the blind trust of each organisation that the translation tool generates error-free smart contracts. Each update of a smart contract requires that each peer must compile, instantiate and validate it before it is committed to the blockchain, thus consuming resources and adding to the overhead of the blockchain’s performance.

It is clear that blockchain is much more expensive as a medium for processing and storage than traditional media. Hence, it should be used as sparingly as possible by minimising both the size of smart contracts and the amount of data stored, while maintaining trust by means of a reliable audit trail. Extraneous processing and data should go to traditional platforms that offer better performance, flexibility and technology heterogeneity, and less visibility across parties. We are not convinced that it is necessary to reinvent the functionality of a BPM engine, which includes complex control flow management, data management and resource allocation, within a blockchain platform.

As we have demonstrated in our implementation, it is less work to integrate blockchain into an application with our federated approach, when compared to the more heavyweight blockchain-centric architectures. To fully transfer all the features of an industrial strength BPM system onto a blockchain platform could amount to a very long, risky and expensive undertaking, especially when considering the non-trivial processes in real-world scenarios. Our prototype illustrates the advantages of dedicating the existing capabilities of BPMS for process automation, and those of blockchain as an immutable, distributed ledger, to automate secure, cross-organisational process interactions without the overheads necessitated by the heavyweight, blockchain-centric approach.

We believe our proposal aligns better with the underlying philosophy of blockchain technology based on distributed autonomous organisations (DAOs) [12]. We have presented a conceptual architecture and an implementation that demonstrates the feasibility of the approach. The comparisons presented here are mostly qualitative; a more thorough empirical comparison through experiments and quantitative data is needed and it will form our future work. More work is also needed to optimise the distribution of on-chain and off-chain data, and to validate the applicability of the federated approach with different types of scenarios and use cases.

References

1. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., et al.: Hyperledger Fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the
2. Astigarraga, T., Chen, X., Chen, Y., Gu, J., et al.: Empowering business-level blockchain users with a rules framework for smart contracts. In: 16th International Conference on Service-Oriented Computing. pp. 111–128. Springer (2018)

3. Buterin, V.: Ethereum: A next-generation smart contract and decentralized application platform (2014). https://github.com/ethereum/wiki/wiki/White-Paper

4. Cachin, C.: Architecture of the Hyperledger Blockchain Fabric. In: Distributed Cryptocurrencies and Consensus Ledgers. vol. 310, p. 4 (2016)

5. Christidis, K., Devetsikiotis, M.: Blockchains and smart contracts for the internet of things. IEEE Access 4, 2292–2303 (2016)

6. García-Bañuelos, L., Ponomarev, A., Dumas, M., Weber, I.: Optimized execution of business processes on blockchain. In: BPM. pp. 130–146. Springer, Cham (2017)

7. Hull, R., Batra, V.S., Chen, Y.M., Deutsch, A., et al.: Towards a shared ledger business collaboration language based on data-aware processes. In: 14th International Conference on Service-Oriented Computing. pp. 18–36. Springer (2016)

8. 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 49(7), 1162–1193 (2019)

9. Mendling, J., Hahner, M.: From inter-organizational workflows to process execution: Generating BPEL from WS-CDL. In: CoopIS’05. pp. 506–515. Springer (2005)

10. Mendling, J., Weber, I., van der Aalst, W., vom Brocke, J., et al.: Blockchains for business process management - challenges and opportunities. ACM Transactions on Management Information Systems 9(1), 4:1–4:16 (Feb 2018)

11. Narendra, N.C., Norta, A., Mahunnah, M., Ma, L., Maggi, F.M.: Sound conflict management and resolution for virtual-enterprise collaborations. Service Oriented Computing and Applications 10(3), 233–251 (Sep 2016)

12. Norta, A.: Creation of smart-contracting collaborations for decentralized autonomous organizations. In: International Conference on Business Informatics Research. pp. 3–17. Springer (2015)

13. Norta, A., Grefen, P., Narendra, N.C.: A reference architecture for managing dynamic inter-organizational business processes. Data & Knowledge Engineering 91, 52–89 (2014)

14. Sturm, C., Szalanczi, J., Schönig, S., Jablonski, S.: A lean architecture for blockchain based decentralized process execution. In: International Conference on Business Process Management Workshops. pp. 361–373. Springer (2018)

15. Szabo, N.: Formalizing and securing relationships on public networks. First Monday 2(9) (1997)

16. ter Hofstede, A., van der Aalst, W., Adams, M., Russell, N. (eds.): Modern Business Process Automation: YAWL and Its Support Environment. Springer (2010)

17. Underwood, S.: Blockchain beyond bitcoin. Communications of the ACM 59(11), 15–17 (2016)

18. Weber, I., Gramoli, V., Ponomarev, A., Staples, M., et al.: On availability for blockchain-based systems. In: 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS). pp. 64–73 (Sept 2017)

19. Weber, I., Xu, X., Riveret, R., Governatori, G., et al.: Untrusted business process monitoring and execution using blockchain. In: BPM. pp. 329–347. Springer (2016)

20. Xu, X., Weber, I., Staples, M., Zhu, L., et al.: A taxonomy of blockchain-based systems for architecture design. In: ICSA. pp. 243–252. IEEE (April 2017)

21. Yasaweeringhegalle, R., Staples, M., Weber, I.: Predicting latency of blockchain-based systems using architectural modelling and simulation. In: ICSA. pp. 253–256. IEEE (April 2017)