CEUR-WS.org/Vol-3045/paper02.pdf

Survey of Model-Driven Engineering Techniques for Blockchain-Based Applications

Olivier Levasseur, Mubashar Iqbal and Raimundas Matulevičius

Institute of Computer Science, University of Tartu, Narva maantee 18, 51009 Tartu, Estonia

Abstract

Blockchain technology is emerging in various domains *(e.g., supply chain, banking, healthcare)* to implement business processes without requiring trusted third parties. However, writing blockchain-based applications (BBAs) is error-prone, and development mistakes are critical since once the smart contract is deployed, its source code can be examined but cannot be modified. Model-driven engineering (MDE) can help overcome these challenges, supporting the secure and fast development of BBAs. In this paper, we utilised the systematic literature review to explore the existing MDE techniques. This study proposes a characterisation of some of the current existing MDE techniques for BBAs that might help in selecting the right modeling technique for building these applications. Our work has revealed several gaps in the current MDE techniques that we addressed in future work.

Keywords

Blockchain, Blockchain model-driven engineering, modeling techniques, Blockchain oracles

1. Introduction

Blockchain technology operates in a trust-less environment and the tamper-proof nature of blockchain gives confidence that data stored on the blockchain will not be tampered by malicious actors [1]. The applications running on top of a blockchain platform are known as blockchain-based applications (BBAs) that have unique characteristics as compared to traditional applications, and building them is error-prone [1, 2]. BBAs operate in a decentralised manner where no third party is needed to ensure that each party behaves according to the defined rules. Also, they enable decentralised inter-organisational cooperation where storing data has a monetary cost and a specific block structure. Furthermore, the development mistakes are critical since once the smart contract (SC) is deployed, the source code cannot be modified [2]. However, the source code can be examined by anyone, and if there exist vulnerabilities, an attacker will exploit them. For example, the decentralised autonomous organisation (DAO) attack allowed the attacker to steal $60 million worth of cryptocurrency [2]. The model-driven engineering (MDE) could support the secure development of BBAs [3], overcome the above-mentioned challenges, and improve the understanding of blockchain systems.

Currently, there exist various modeling techniques to model BBAs and automatically generate SCs code by using different modeling languages; for instance, domain-specific languages [4],

PoEM'21 Forum: 14th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modelling, November 24-26, 2021
olivier.levasseur@ut.ee (O. Levasseur); mubashar.iqbal@ut.ee (M. Iqbal); rma@ut.ee (R. Matulevičius)
0000-0003-0543-613X (M. Iqbal); 0000-0002-1829-4794 (R. Matulevičius)
© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

state machines [3], or business processes [5]. A few also provide intuitive user interfaces that support the development life-cycle of the application [5]. However, the extent of modeling and MDE for BBAs is not defined explicitly, which makes it difficult to determine the current state of MDE for BBAs, what are the limitations and research opportunities. In this paper, we follow the systematic literature review (SLR) [6] to identify the available modeling techniques and summarise their characteristics and modeled elements (including off-chain elements, e.g., oracles) required for designing BBAs. In this paper, smart contracts and blockchain-based applications are used interchangeably.

The paper is structured as follows: Section 2 discusses the background. Section 3 presents the research method and explains the survey settings. Section 4 elaborates the results. Section 5 describes the future work, threats to validity and concludes the paper.

2. Background

2.1. Blockchain

The advent of bitcoin cryptocurrency (the first blockchain implementation) brings the concept of blockchain [7]. Blockchain operates over a peer-to-peer (P2P) network, performs transactions in decentralised manners, and connects each block to the previous block by a unique cryptographic hash [8]. Every block contains a valid list of transactions that are hashed and ordered as a Merkle tree [9]. Existing blockchains can be classified as permissionless (e.g., bitcoin, ethereum) or permissioned blockchains (e.g., hyperledger fabric) [5]. In permissionless blockchains, anyone can join the network and view the ledger. In contrast, permissioned blockchains have pre-verified nodes and require certain permissions to view and execute transactions.

The emergence of SCs, which represent source code running on top of blockchain platforms, allows developers to write decentralised applications in various domains (e.g., supply chain, banking, healthcare). The characteristics of blockchain platforms enable inter-organisational cooperation in a trust-less environment where no third party is needed to ensure that each party behaves according to the defined rules [10]. Additionally, the immutable ledger keeps traces of each party's operations that support monitoring the process and auditing.

2.2. Model-driven Engineering

Model-driven engineering is a software development methodology that uses domain models to discover solutions in a domain. MDE can be useful in translating a platform-independent model (e.g., ArchiMate model) to a platform-specific language or model (e.g., Hyperledger composer model) [11], abstracting the unique characteristics of blockchain platforms, to generate source code automatically or to translate a model to another one, and making application development secure and faster. For example, ChorChain [10] allows stakeholders to model, compile, and deploy the SCs on the Ethereum blockchain. As a result, MDE automates the development life-cycle and supports the faster development of BBAs.

MDE can also provide an additional level of security, especially for permissionless blockchains since anyone on the network can interact with them, potentially exposing their vulnerabilities [12]. In MDE, modeling techniques *(e.g., Business Process Model and Notation (BPMN), Unified*

Modeling Language (UML), Flow Charts) can help to design secure BBAs by integrating secure design patterns in the modeling process and promoting the best development practices. For instance, Blockchain Studio integrates access control policies into a BPMN model [8]. As another example, FSolidM offers an access control extension, protection against reentrancy, and state unpredictability which are vulnerabilities of Ethereum SCs [3].

2.3. Related Work

We argue that MDE can help with complex design decisions, including complex computations, storing data, keeping data private, querying on/off-chain data, and error-free code generation. Here, we discuss a few papers relevant to the MDE for BBAs.

Wohrer et al., [13] provide the architectural design decisions to overcome the challenges in designing BBAs. The study follows the grounded theory approach to collect, extract, and summarise the common practices in design options for blockchain-based solutions.

Falazi et al., [14] discuss modeling techniques that lack to model fine-grained decisions when handling the uncertainty of blockchain transactions or when off-chain communicates with on-chain. The authors use the blockchain-aware modeling and execution (BlockME) method and present a modeling extension to model the above-mentioned interactions.

Xu et al., [15] present originChain that provides structural design of SCs and its analysis based on adaptability, upgradability, the latency of writing and reading. Markovska [16] investigates the blockchain-oriented processes that can be modeled with the activity-centric modeling paradigm of BPMN. Udokwu et al., [17] provide the decentralised agent-oriented modeling (DAOM) framework and a support tool for designing BBAs.

The related studies debate the specific modeling technique or aspect of BBAs and none of them looked into the various constructs and modeling approaches to model BBAs.

3. Research Method

We follow the SLR method [6] to explore the existing MDE (and modeling) techniques to summarise the constructs used to model BBAs. In other words, the techniques that model the architecture of BBAs, their latency or resource consumption are excluded. To achieve the research objective of this work, we formulated three research questions.

[RQ1] What are the main characteristics of existing modeling techniques?
The ***RQ1*** focuses on general characteristics of the modeling techniques. For example, i) What are the blockchain type and modeling language? ii) Does the model allow formal verification? iii) Whether the model allows automatic code generation or not? iv) Is data privacy supported? v) Is life-cycle management supported? vi) Whether the technique is open source or not?

[RQ2] What are the elements of blockchain-based applications that are modeled in existing modeling techniques?
Some existing modeling techniques represent BBAs as a state machine and other model access control, or SCs. The objective of ***RQ2*** is to identify and enumerate those constructs and to spot

where improvements are needed.

[RQ3] What types of oracles are modeled and how are they modeled?
BBAs are needed a way to access data that is not stored on-chain. The oracles connect off-chain communication to BBAs. The ***RQ3*** helps us to explore the types of oracles and how they are modeled.

To conduct SLR, a predefined **review protocol** was established to overcome the biases of researchers and to develop a seamless process to perform a SLR [6]. The **selection of electronic databases** and literature search is carried out after consultation with blockchain domain specialists. The initial search for relevant literature studies was done through the ACM digital library, the IEEE digital library, ScienceDirect, and Scopus. The **search terms** *(("blockchain" OR "blockchain-based application") AND ("modeling" OR "modelling" OR "model-driven engineering")) AND ("techniques" OR "methods" OR "strategies")* were formulated based on the research scope to collect relevant literature studies. The **inclusion and exclusion criteria** is listed in Table 1.

Table 1
Inclusion and exclusion criteria

Inclusion criteria	Exclusion criteria
Papers related to MDE of blockchain applications	Papers published before 2008 or not available freely
Papers use modeling techniques for blockchain applications	Papers shorter than 5 pages or not written in English

As a first step, we run predefined search terms on selected digital databases to collect **relevant literature studies**. The corpora of search results were downloaded as BibTeX files and a python script was written to filter the results. The python script verified that the predefined search terms are available in the **title** or **abstract** of the study. After running the script on the BibTeX files, we collected a total of 236 papers for further screening. The screening was made after reading the paper title, abstract, introduction, conclusion, and filtering based on the inclusion and exclusion criteria. Finally, 8 literature studies remained, from which we discovered additional 7 studies. Thus, we collected **15 relevant literature studies** for full-text reading and analysis.

4. Results

In this section, we summarise the results of the SLR. Table 2 shows the selected literature studies, their contribution, and limitations.

Rocha et al., "Preliminary steps towards modeling blockchain oriented software" [18]
This article proposes no novel MDE technique but shows how existing methods could be used to model blockchain-based applications. More specifically, UML, BPMN, and ER models are illustrated to model the blockchain elements. The objective was to show how the existing approaches lack certain concepts to cover different blockchain elements.

Mavridou et al., "Designing secure Ethereum smart contracts: A Finite State ..." [3]
The authors model SCs as finite state machines through a graphical editor. The idea is to use rigorous semantics to lay the foundation for formal verification tools. SC (written in solidity

Table 2
Summary of selected relevant literature studies

	Modeling tech.	**Contribution**	**Limitation**
Rocha et al., [18]	UML, BPMN, ERD	Explore current modeling languages	Not an extension of a language
Mavridou et al., [3]	WebGME	SC Modeling, security extensions, formal modeling	Limited modeled elements, no off-chain modeling
Lopez-Pintado et al., [5]	BPMN	On-chain BPMS and modeled elements	No access control, no full support of oracles
Mercenne et al., [8]	BPMN	Extension of caterpillar with access control	No full support of oracles
Silva et al., [19]	DEMO	Meta-model of DEMO and HLF concepts	No code generation, no off-chain components modeling
Hornkov et al., [20]	DEMO	Highlights importance of interaction with off-chain	No code generation, no off-chain components modeling
Ladleif et al., [21]	BPMN Choreography	Extension of BPMN 2.0 choreography diagrams	No modeling of , escrow, or oracles
Corradini et al., [10]	BPMN Choreography	Translation of BPMN choreography to SC, dApp life-cycle	No full support of oracles
Weber et al., [22]	BPMN Choreography	Triggers as a communication method for on- and off-chain	Triggers not modeled, no full support of oracles
Marchesi et al., [7]	UML	Extension of UML, agile dApp development method	No code generation, no full support of oracles
Hamdaqa et al., [4]	Domain specific lang.	Unified reference model for SC of multiple blockchains	No code generation, no off-chain components modeling
Garamvolgyi et al., [23]	UML Statecharts	modeling of SC as a state machine	No code generation, no off-chain components modeling
Lu et al., [12]	BPMN	SC interfaces and tokens model, off-chain communication	No access control policies, no full support of oracles
Babkin et al., [11]	ArchiMate	Automatic translation between Archimate and HLC	Manual work before SC deployment
Boubeta-Puig et al., [24]	BPMN and EMF	Integration of CEP with Ethereum Platform	No full support of oracles

code) can be generated automatically from the model. The model also allows developers to easily integrate security design patterns (e.g., access control) into the SCs.

Lopez-Pintado et al., "Caterpillar: a business process execution engine ..." [5] and Mercenne et al., "Blockchain Studio: A Role-Based Business Workflows ..." [8]
Both articles support business processes where the study [5] provides a BPMN-compatible business process management system (BPMS) on Ethereum. The BPMN model is translated to multiple SCs that allow the execution and management of the business process. The tool comes with a user interface that facilitates the life-cycle management of the processes. Mercenne et al., [8] create a fork of Caterpillar to support the management of roles and access control.

Silva et al., "Decentralized Enforcement of Business Process Control Using Blockchain" [19] and Hornkov et al., "Exploring a Role of Blockchain Smart Contracts ..." [20]
Both of these techniques use the enterprise ontology DEMO to model blockchain-based applications. Silva et al., [19] propose a translation between DEMO and Hyperledger. The study improves the control and traceability of collaborative business processes. The method [20] presents an enterprise information system to support the transactions that do not need to be stored on the blockchain. The presented method does not offer automatic code generation.

Ladleif et al., "Modeling and Enforcing Blockchain-Based Choreographies" [21] , Corradini et al., "Engineering trustable choreography-based systems using blockchain" [10] and Weber et al., "Untrusted Business Process Monitoring and Execution ..." [22]
Ladleif et al., [21] extend BPMN choreographies to support business processes on Ethereum. Choreography diagrams look at the system from a message exchange perspective. Similarly, Corradini et al., [10] present a method that uses BPMN choreography diagrams to model blockchain-based applications. The study [22] also uses choreography diagrams to ensure that only confirming messages advance the state of the process.

Marchesi et al., "An Agile Software Engineering Method to Design Blockchain ..." [7]
The paper suggests an agile method that separates the development of an application (e.g., SCs and traditional systems). It considers various design stages, like the user interface to the external SCs, emphasises the security concerns of traditional and blockchain-based applications.

Hamdaqa et al., "iContractML" [4]
The study enables a feature-oriented domain analysis of three blockchain platforms (Hyperledger, Azure, and Ethereum) to build a reference model for SCs and presents a platform-independent modeling language called iContractML that allows automatic code generation.

Garamvolgyi et al., "Towards Model-Driven Engineering of Smart Contracts ..." [23]
This article explores the use of UML state charts to model cyber-physical systems (CPS). The logic is implemented as a SC and is used as a digital twin to the real-life CPS. The approach is meant to be generic enough to allow generating code for different blockchain platforms.

Lu et al., "Integrated model-driven engineering of blockchain applications for business processes and asset management" [12]
In this article, the authors present a tool called Lorikeet that allows the modeling of business processes and digital assets. They allow the creation of both fungible and non-fungible assets, escrow, and assets swap. The tool comes with a user interface to facilitate the modeling and interaction with the blockchain-based application.

Babkin et al., "Model-Driven Liaison of Organisation Modeling Approaches ..." [11]
In this paper, the authors present a mapping between concepts of ArchiMate enterprise architecture modeling language and the HyperLedger composer modeling language. The implementation of the automatic code generation is done in Python, and the code translation is done semi-automatically, i.e. the business logic must be manually written.

Boubeta-Puig et al., "CEPChain: A graphical model-driven solution for integrating complex event processing and blockchain" [24]
CEPchain connects a complex event processing system to the Ethereum platform to allow the modeling of the automatic trigger of SCs transactions when event pattern conditions are met.

4.1. Characteristics of Modeling Techniques

In characteristics of the modeling techniques (Table 3), it can be seen that there are techniques that allow source code generation from the model. A few techniques also allow life-cycle management, meaning that they help the developer with the compilation and deployment of the SCs and interaction with the SCs through user interfaces.

However, most of the examined modeling techniques do not allow data privacy when dealing with permissionless blockchains, and it is a limitation imposed by the nature of the platform. None of the techniques allows formal verification that can be useful to verify a model against a set of validation rules.

Table 3
Characteristics of modeling techniques

	Blockchain platform	modeling language	Formal verification	Code generation	Data privacy	Life-cycle management	Open source
Rocha et al., [18]	Ethereum	BPMN, ER and UML	No	No	No	No	No
Mavridou et al., [3]	Ethereum	FSolidM	No	Full	No	No	Yes
Lopez-Pintado et al., [5]	Ethereum	BPMN	No	Full	No	Yes	Yes
Mercenne et al., [8]	Ethereum	BPMN	No	Partial	No	Yes	No
Silva et al., [19]	Hyperledger	DEMO	No	No	Yes	No	No
Hornkov et al., [20]	Ethereum	DEMO	No	No	No	No	Yes
Ladleif et al., [21]	Ethereum	BPMN choreography	No	Full	No	No	No
Corradini et al., [10]	Ethereum	BPMN choreography	No	Full	No	Yes	Yes
Weber et al., [22]	Ethereum	BPMN choreography	No	Full	Yes	No	No
Marchesi et al., [7]	Ethereum	UML	No	No	No	No	No
Hamdaqa et al., [4]	Azure, Ethereum, Hyperledger	DSL (eCore)	Model validation rules	Partial	No	No	Yes
Garamvolgyi et al., [23]	Ethereum	UML state charts	No	Partial	No	No	No
Lu et al., [12]	Ethereum	BPMN	No	Full	No	Yes	No
Babkin et al., [11]	Hyperledger	Archimate	No	Partial	Yes	No	No
Boubeta-Puig et al., [24]	Ethereum	BPMN and EMF	No	Full	No	Yes	No

4.2. Modeled Elements of Blockchain-based Applications

The blockchain elements (Table 4) are ***1) SC rules*** that generate the function bodies of SCs and determine the behaviour of the application. ***2) SC data***, for example, the type and the value of the information that is stored on the blockchain. ***3) 3rd party SC*** that are not directly part of the modeled blockchain application. It could be any SC that existed before the application was modeled. ***4) Event*** is emitted on the blockchain that other actors can subscribe to. ***5) Transaction*** is an action that modifies the state of the blockchain and/or the state of the application. ***6) Participant*** is an actor that interacts with the application. ***7) Roles and permissions*** are restrictions to specific users. For instance, not everyone should be allowed to empty the balance of a SC. ***8) Asset*** has a value that can be exchanged (e.g., tokens).

The SC logic and business rules are the most examined and well modeled by the different modeling techniques. The data stored in the SC, the roles, permissions, and transactions also seem to be relatively well modeled. On the other hand, the interaction with third party SCs is rarely modeled. This is a limitation since SCs often need to access data from other contracts on the blockchain. Additionally, the assets on the blockchain are poorly represented by existing modeling techniques, especially by those which provide code generation features. Finally, blockchain events are also neglected by the existing modeling techniques.

4.3. Modeled Oracles

BBAs need to access data that is not stored on-chain. The components that provide external data to blockchain are called oracles. Oracles can be characterised by the direction of the data flow (inbound or outbound) and they can also be characterised by the initiator of the data flow (pull-based or push-based) [25].

In **pull-based inbound**, the initiator is a SC, and the data flow is from off-chain to blockchain. The off-chain subscribes to events emitted by the on-chain application. Once the oracle detects an event, it will fetch the data and send a signed transaction containing the data to the requesting on-chain application. In **push-based inbound**, the initiator is off-chain, and the data flow is from off-chain to blockchain. Here, an off-chain component is configured to send data under certain conditions, which is then translated into a blockchain transaction.

Table 4
Modeled elements of blockchain-based applications based on the identified relevant works. (X) indicates that an element is not modeled.

	Rules	SC data	3rd party SC	Event	Transaction	Participant	Roles	Asset
Rocha et al., [18]	ER, UML, BPMN	Database columns, Class attributes	X	X	UML class function, association	UML: SC class, BPMN: lane	X	Database column, Class attribute
Mavridou et al., [3]	X	Variables in code editor	X	X	State transition	X	Guards	Variables in code editor
Lopez-Pintado et al., [5]	BPMN scripts and control flow	X	X	BPMN events	BPMN tasks	BPMN lanes	X	X
Mercenne et al., [8]	BPMN scripts and control flow	X	X	BPMN events	BPMN tasks	BPMN lanes	BPMN task role attribute	X
Silva et al., [19]	X	DEMO Fact	X	X	Business transaction	DEMO actor	DEMO actor role	Busines transaction
Hornkov et al., [20]	DEMO actions	Transaction object facts	X	X	DEMO transaction	DEMO actor	DEMO actor	X
Ladleif et al., [21]	BPMN script tasks	BPMN data objects	Script tasks	X	Sequence flow	Initiator and recipient	Sequence flow	X
Corradini et al., [10]	Gateways, guards, sequence flows	Partly represented by message annotations	X	Start event, end event	Tasks (messages)	Initiator and recipient of messages	Checkbox to create choreography instance	Message annotations
Weber et al., [22]	BPMN choreography	X	X	BPMN events	Choreography messages	Lanes	Lanes, sender/ receiver	X
Marchesi et al., [7]	UML classes and stereotypes	UML class attributes and stereotypes	UML class attributes and stereotypes	UML stereotype	UML sequence	UML stereotype	UML stereotype	Message in UML sequence
Hamdaqa et al., [4]	UML classes	UML classes	X	UML class	UML classes	UML class	UML classes	UML class
Garamvolgyi et al., [23]	Transition, guards	States, history	X	X	Action, Transition	X	Transacti. guard	X
Lu et al., [12]	BPMN scripts	Script tasks, asset templ.	SC Interface icon	BPMN events, UI event	BPMN task	User tasks	X	Asset template forms
Babkin et al., [11]	X	Artifact, Business object, Data object	X	Application-, Business-, Implementation-, Technology event	Application-, Business-, and Technology-Interaction	BusinessActor, Stakeholder, Application-Component	Access Relationship	Same as smart contract data
Boubeta-Puig et al., [24]	BPMN scripts and control flow	X	X	BPMN events	BPMN tasks	BPMN lanes	X	X

In **pull-based outbound**, the initiator is an off-chain, and the data flow is from blockchain to off-chain. In **push-based outbound**, the initiator is on-chain and the data flow is from blockchain to off-chain.

Table 5
Oracles modeling coverage

	Pull-based (inbound)	Push-based (inbound)	Pull-based (outbound)	Push-based (outbound)
Rocha et al., [18]	Yes	Yes	Yes	Yes
Lopez-Pintado et al., [5]	Partly modeled through events	No	No	Partly modeled through events
Weber et al., [22]	Partly modeled through events	No	No	Partly modeled through events
Marchesi et al., [7]	Partly modeled through events	No	No	Partly modeled through events
Lu et al., [12]	Partly modeled through events	No	No	Partly modeled through events
Boubeta-Puig et al., [24]	No	Yes	No	No

In a few techniques, there have been attempts at modeling pull-based inbound and push-based outbound oracles (Table 5), but no modeling technique supports automatic code generation for all types of oracles (Table 6). For the push-based inbound and pull-based outbound oracles, user interfaces are nice but they are not scalable, because the actions must be done manually. According to the literature review, the automatic code generation of push-based inbound and pull-based outbound oracles is supported through user interfaces and REST APIs (Table 6).

Table 6
Automatic code generation coverage of oracles

	Pull-based (inbound)	Push-based (inbound)	Pull-based (outbound)	Push-based (outbound)
Lopez-Pintado et al., [5]	No	Partly through API	Yes through API	No
Ladleif et al., [21]	No	Partly through custom interface	Yes through custom interface	No
Corradini et al., [10]	No	Partly through UI	Yes through UI	No
Babkin et al., [11]	No	Partly through API	Yes through API	No
Lu et al., [12]	No	Partly through API	Yes through API	No
Boubeta-Puig et al., [24]	No	Yes	No	No

5. Discussion and Concluding Remarks

There are a few limitations to our current work that we discussed as **threats to validity** [26]. The **restricted time span** refers to the researchers' incapacity to foresee relevant studies outside of the time frame planned during the SLR planning phase. In addition, some of the modeling techniques may have been modified or improved between the time of this work. This limitation is a reality of SLRs and cannot really be mitigated. The **bias in study selection** occurs when researchers have their own subjective conjecture and do not apply inclusion and exclusion criteria consistently or use incompatible search terms. An attempt to counter this limitation is made by gathering feedback from other researchers in the field to include any missing relevant work in the SLR. The **bias in data extraction** and **subjective interpretation** arise when researchers have different interpretations and opinions about the extracted data. To mitigate these biases, researchers involved in the SLR to share their point of view and discuss until they reach a consensus.

In this work, we present an analysis of model-driven engineering techniques for blockchain-based applications. As a result, we summarise the current state of MDE and propose a characterisation that enables the iteration on various modeling techniques based on their characteristics and modeled elements. In **future work**, the existing modeling techniques can be extended to cover blockchain oracles modeling and automatic code generation. Another possible research direction is to provide security extensions in modeling techniques to take advantage of the security practices when designing BBAs.

Overall, completing the above-mentioned future work and tackling the threats to validity could provide better insights and in-depth contributions to the MDE of BBAs.

References

[1] M. Iqbal, R. Matulevičius, Comparison of Blockchain-Based Solutions to Mitigate Data Tampering Security Risk (2019) 13–28.

[2] N. Atzei, M. Bartoletti, T. Cimoli, A survey of attacks on Ethereum smart contracts (2017).

[3] A. Mavridou, A. Laszka, Designing secure ethereum smart contracts: A finite state machine based approach, in: Financial Cryptography and Data Security, 2018, pp. 523–540.

[4] M. Hamdaqa, L. A. P. Metz, I. Qasse, iContractML, in: 12th System Analysis and Modelling Conference, 2020.

[5] O. López-Pintado, L. García-Bañuelos, M. Dumas, I. Weber, A. Ponomarev, Caterpillar: A business process execution engine on the ethereum blockchain (2019).

[6] B. Kitchenham, S. Charters, Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3, Engineering 45 (2007) 1051.
[7] M. Marchesi, L. Marchesi, R. Tonelli, An agile software engineering method to design blockchain applications, in: 14th CEE-SECR, 2018.
[8] L. Mercenne, K.-L. Brousmiche, E. B. Hamida, Blockchain studio: A role-based business workflows management system, in: IEEE 9th Annual IEMCON, 2018.
[9] M. Iqbal, R. Matulevičius, Blockchain-Based Application Security Risks: A Systematic Literature Review, Springer Nature Switzerland AG (2019) 1–26.
[10] F. Corradini, A. Marcelletti, A. Morichetta, A. Polini, B. Re, F. Tiezzi, Engineering trustable choreography-based systems using blockchain, in: Symposium on Applied Comp., 2020.
[11] E. Babkin, N. Komleva, Model-driven liaison of organization modeling approaches and blockchain platforms, in: Advances in Enterprise Engineering XIII, 2020, pp. 167–186.
[12] Q. Lu, A. B. Tran, I. Weber, H. O'Connor, P. Rimba, X. Xu, M. Staples, L. Zhu, R. Jeffery, Integrated model-driven engineering of blockchain applications for business processes and asset management, Software: Practice and Experience 51 (2020) 1059–1079.
[13] M. Wohrer, U. Zdun, Architectural design decisions for blockchain-based applications, IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2021 (2021).
[14] G. Falazi, M. Hahn, U. Breitenbücher, F. Leymann, Modeling and execution of blockchain-aware business processes, Software-Intensive Cyber-Physical Systems 34 (2019) 105–116.
[15] X. Xu, Q. Lu, Y. Liu, L. Zhu, H. Yao, A. V. Vasilakos, Designing blockchain-based applications a case study for imported product traceability, FGCS 92 (2019) 399–406.
[16] M. Markovska, Modelling Business Processes on a Blockchain Ecosystem (2019).
[17] C. Udokwu, P. Brandtner, A. Norta, A. Kormiltsyn, R. Matulevičius, Implementation and evaluation of the DAOM framework and support tool for designing blockchain decentralized applications (2020) 0–19.
[18] H. Rocha, S. Ducasse, Preliminary steps towards modeling blockchain oriented software, in: 1st Workshop on Emerging Trends in Software Engineering for Blockchain, 2018.
[19] D. Silva, S. Guerreiro, P. Sousa, Decentralized enforcement of business process control using blockchain, in: Advances in Enterprise Engineering XII, 2018, pp. 69–87.
[20] B. Hornáčková, M. Skotnica, R. Pergl, Exploring a role of blockchain smart contracts in enterprise engineering, in: Advances in Enterprise Engineering XII, 2018, pp. 113–127.
[21] J. Ladleif, M. Weske, I. Weber, Modeling and enforcing blockchain-based choreographies, 2019, pp. 69–85.
[22] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, J. Mendling, Untrusted business process monitoring and execution using blockchain, 2016, pp. 329–347.
[23] P. Garamvolgyi, I. Kocsis, B. Gehl, A. Klenik, Towards model-driven engineering of smart contracts for cyber-physical systems, in: 48th annual IEEE/IFIP DSN-Workshops, 2018.
[24] J. Boubeta-Puig, J. Rosa-Bilbao, J. Mendling, CEPchain: A graphical model-driven solution for integrating complex event processing and blockchain, Expert Systems with Applications 184 (2021) 427–435.
[25] R. Mühlberger, S. Bachhofner, E. C. Ferrer, C. D. Ciccio, I. Weber, M. Wöhrer, U. Zdun, Foundational oracle patterns: Connecting blockchain to the off-chain world, 2020.
[26] X. Zhou, Y. Jin, H. Zhang, S. Li, X. Huang, A map of threats to validity of systematic literature reviews in software engineering, in: APSEC (2016) 153–160.