Comparing Blockchain Platforms for Smart Contracts: A Preliminary Framework

Giuseppe Di Lucca and Maria Tortorella

August 30, 2021

Comparing Blockchain Platforms for Smart Contracts: A Preliminary Framework

Giuseppe A. Di Lucca, Maria Tortorella
Dept. of Engineering
University of Sannio, Benevento, Italy
dilucca@unisannio.it, tortorella@unisannio.it

Abstract

Since 2008, when it was first cited, blockchain technology represents an innovation from both a structural and application point of view. Since then, thanks to its peculiarities and capabilities of implementing smart contracts, blockchain technology has undergone a strong development in different application domains. The interest around this technology also brought to the definition of several platforms facilitating its use and application. Due to their variety, , choosing the most suitable blockchain platform to support a specific business need represents a strategic problem. This paper proposes an analysis for the definition of an evaluation framework and related quality attributes, helping to characterize and compare different blockchain platforms for identifying the most suitable one to the implementation of smart contracts in a specific business context. The analysis of a set of blockchain platforms is proposed for discussing the applicability and use of the proposed framework

1 Introduction

Blockchain technology has achieved more and more interests in the last years. It was proposed for supporting and managing exchange and payment systems based on cryptocurrencies [20], largely influencing the financial industry. Taking leverage of its main advantages, such as decentralization, immutability, and transparency [22], from the beginning, it has been exploited to develop a large variety of other applications, for different use cases and application domains, such as financial services, chain supply, IoT, banking, manufacturing, and so on, and transformed much in the way common everyday objects (e.g., smartphones, cars) and personal data (e.g., personal identification, healthcare) are used [6].

The increasing interest around blockchains has led to the definition of several platforms easing their implementation and application, in particular in the execution of smart contracts [21]. A smart contract is a program executed on a blockchain that allows for automatic negotiation and agreement among multiple entities/parties.

Due to the large variety of use cases and application domains, a specific blockchain technology may not be able to meet the requirements for all use scenarios. Therefore, choosing the most suitable blockchain platform to execute smart contracts in a specific business context is a strategic issue. Each platform has its own features that makes it more or less suitable to be used for specific needs.

This paper proposes a framework with a set of quality attributes to help characterize blockchain platforms, analyze them and identify the most suitable to meet specific business needs. The aim of the framework is twofold: supporting a software engineer in formalizing its perceived quality with reference to a blockchain platform that is to be used to implement a smart contract, and understand how attractive the platform is for the software engineer.

The framework only considers external quality attributes. In addition to previous studies regarding software quality assessment, the definition of the framework took into consideration the existing blockchain literature and websites/blogs dealing with similar issues. To the knowledge of the authors, there are no quality assessment frameworks regarding blockchains in the literature and this paper is intended as a first attempt to define it. A case study has been carried out to verify the applicability and usefulness of the proposed framework for analyzing blockchain platforms.

The paper is structured as follows. Section II reports some backgrounds regarding basic information and concepts on blockchains and smart contracts. Section III discusses related work. In section IV, the proposed quality framework is presented, while section V describes the application of the framework in a case study. Finally, Section VI includes conclusive remarks.

2 Backgrounds

A blockchain is a persistent, shared and immutable data structure composed of a sequence (i.e., a chain) of records (i.e., blocks) chronologically linked to each other. Each block is linked by a hash to the previous one and has its own time stamp. Once data (i.e., transactions) are written/registered in a block, they cannot be modified or deleted, otherwise the whole blockchain will be invalid, as a change in a block would require modifying all its subsequent blocks. Cryptography is used to guarantee integrity and security of data recorded in a block. Thus a blockchain makes up a distributed digital ledger to store transactions in blocks in a permanent, verifiable and secure way. Blocks are added by miners, computational nodes distributed on the network forming the blockchains. To add a new block, a predefined specific validation process, said Consensus, is performed to ensure a validation of the block and a correct chaining of transactions and provide a guaranteed unique order. It involves miners and/or other nodes. Many Consensus methods have been defined [28]. The main consensus methods are the Proof of Work (PoW) and the Proof of Stake (PoS). PoW requires to solve (cryptographic) puzzles (or, more generally, a very computationally complex mathematical function) made up by all information previously recorded on the blockchain (that nodes will 'mine') and by a new set of transactions to be added to the next block: the (group of) nodes that first will solve the puzzle/function have the rights to create the new block. PoS requires a (group of) node(s) to prove the ownership of a certain amount of token (e.g., coins of some cryptocurrencies, or other assets) to participate in the validation of transactions. The node(s) with more tokens will have a higher chance to be the validator of the next block (i.e., the seniority of the node(s) is greater than the other nodes, assigning it/them a trusted reputation). However, other consensus mechanisms exist, such as Proof of Importance (PoI), and Byzantine Fault Tolerance (BFT).

A blockchain can be Permissionless or Permissioned. Permissionless blockchains allow anyone to have access to the blockchain and no approval from any authority is required; there is no central owner/control of the network and software. Permissioned blockchains require that an administrator will approve the access to the network, thus transaction validators are pre-selected by blockchain administrators. A permissioned blockchain can be a public, or open, one (i.e., a blockchain which anyone can access and view, but where only authorised participants are permissioned to generate transactions and/or update its state), or closed one. Permissioned closed blockchains are usually enterprise blockchains with restricted access and only the administrator is permissioned to generate transactions and/or update the state.

A blockchain can rely on its own cryptocurrency or to operate and manage different cryptocurrencies also by interacting with other blockchains.

Blockchains were born to support and manage exchange and payment systems based on cryptocurrencies, but in recent years they have been exploited to develop a large variety of other applications, for several use cases and application domains, such as financial services, chain supply, IoT, banking, manufacturing. The introduction of smart contracts [1, 2, 3, 4] has highly leveraged the adoption of blockchain technologies in these domains.

A smart contract can represent a digital agreement/transaction among parties/entities, usually in a business process, as done by a traditional paper contract. With respect to a traditional contract, smart contracts have the advantage that the involved parties/entities are able to codify their agreements/transactions in an automated way without the need of a central authority supervision.

In [5], a smart contract is defined as " … a collection of code and data (sometimes referred to as functions and state) that is deployed using cryptographically signed transactions on the blockchain network … The smart contract is executed by nodes within the blockchain network; all nodes that execute the smart contract must derive the same results from the execution, and the results of execution are recorded on the blockchain".

Smart contracts do not have to perform just business functions: they can perform any kind of calculations, or store information, or expose properties and so on. Smart contracts, being executed on the top of a blockchain, benefit/share blockchain characteristics: e.g., they are tamper evident and tamper resistant (on each node of the blockchain there is a copy of the contract, to avoid tampering); they have to be deterministic; all the nodes executing the contract must agree on the new state after the execution according the adopted consensus method.

Smart contracts can be coded by some traditional programming languages (such as Java, C++, Javascript, Go) or by languages specific for the blockchain hosting them (e.g., Solidity). However, there are blockchains that do not allow the execution of smart contracts. There are blockchains, usually permissionless ones, requiring the user to pay for the cost of the contract execution. Usually there are limitations to contract execution time and the execution is stopped when the limit is exceeded, and the running transaction discarded.

3 Related Work

Papers dealing with quality issues of blockchains are concerned to their technical and performance aspect, while, to the best of authors' knowledge, no papers have dealt with the attributes proposed in this paper. In the following, some works about quality aspects of blockchains are reported and shortly discussed.

In [6], Kahn et al. conduct a comprehensive survey of the literature on blockchain-enabled smart contracts to identify research areas that need further studies. The focus was to analyse smart contracts from both technical point of view (e.g., coding, security, performance issues) and usage point of view (e.g., smart contract applications in finance, healthcare, etc). The main results were: (i) the proposal of a taxonomy of existing blockchain-enabled smart contract solutions with respect to two categories: smart contract improvement and smart contract usage; (ii) categorization of the papers included in the survey; (iii) the identification of a set of current challenges and open issues to be addressed by future work. The analysed features concern Programming-centric solutions, Formal Verification, Security optimization, from the technical/improvement point of view; while they considered: Data and Device Management, Cloud related, Profit or Non-Profit centric solutions, from the Usage point of view.

In [7], the authors present a tradeoff matrix that corresponds to the tradeoffs among the major software architecture quality attributes in evolving blockchain solutions. The authors propose to use the matrix to support the architectural design of software systems comprising a blockchain, so that key attributes are enhanced, or alternatively, de-scoped, in the design of a new blockchain architecture. The authors consider the following attributes: Availability, Modifiability, Testability, Interoperability, Security, Performance, Usability, Extensibility. They use a tool and simulation to get the best tradeoff among the attributes considered with respect to some specification.

In [8], the authors conducted a literature review to analyze the quality issues in the blockchain implementation and to identify the blockchain quality attributes. The research has shown that many studies have focused on one or more particular aspects related to blockchain implementation, but that the research on quality requirements for blockchain implementation is still in its early stage. The most common blockchain quality issues found in the literature were related to security, privacy, throughput, size and bandwidth, performance, usability, data integrity and scalability, setting up a lot of challenges that need to be addressed.

The framework proposed in this paper considers the works above but considers additional quality attributes.

Websites and blogs also provide the evaluations and ranking of blockchain platforms, according to some specified set of attributes. They have been selected by querying Google with “most used blockchain platforms for smart contracts” and selecting the sites that reported also the list of the features considered to compute the ranking. The columns of Table 1 indicate the analysed websites and blogs, while all the attributes they consider are listed on the rows of the table. The table indicates that a wide variety of characteristics are considered and shows that their overlap with reference to the sites analyzing them is low. The table also highlights that the sites adopt a different the approach to the evaluation and there are sites, such as those labeled Hackernoon and LeewayHertz, that are more oriented to evaluate the organization producing and/or supporting the technology, and other ones, such as those labelled Itransition and Gartner, that are more interested to analyse the platform use and performance. G2 is the richest in terms of analysed features. Features concerning Type of Permission, Consensus, transaction speed, governance, transaction costs, and supported languages are common among the considered sites. Features related to security, interoperability, scalability, technical support, are also analysed.

This paper analyses the works discussed above and considers the features proposed in Table 1, together with additional ones, and groups and organizes them for defining the proposed framework.

4 The Proposed Framework

To select a blockchain platform for identifying the most suitable to implement the smart contracts for a specific application domain or use case cannot be an easy task.

Considering the current literature on blockchain quality [6, 7, 8, 21] and the analysis reported in Table 1, a set of attributes that best characterize a blockchain platform have been identified. It is a comprehensive set covering all those aspects/features of interest for a software engineer that has to select a platform that meets specified requirements to execute smart contracts for a specific business context.

The set of attributes has to be defined so that a software engineer can identify which platform (i) shows the higher perceived quality, and (ii) is the most attractive in the specific business context.
A product is perceived to be of good quality when it meets specified requirements about the features of interest for its users. Then, a set of quality attributes have been defined to identify the expected

peculiarities of a blockchain platform and to highlight objective differences among different platforms. To define the attributes for the perceived quality, four main features characterizing a blockchain have been considered: ledger, permissions, smart contracts and consensus. Thus, attributes regarding: permission and consensus policies, smart contracts execution performance, aspects of scalability, complexity, waiting time and ledger management, and registration of the executed transactions, have been defined. More attributes capturing the ease of use of the platform have been considered too, as the ones concerning the used programming language or supporting operative environment.

	Itransition [23]	**G2 [24]**	**LeewayHertz - Software Development Company [25]**	**Gartner peerinsights [26]**	**Hackernoon [27]**
Execution environment	X				
Smart contract	language	X	X		X
Turing completeness	X				
Permission type	X		X		Type of network
Meets Requirements		X			
Easeness		Use Setup Admin		Use Deployment, Integration using Standard APIs and Tools	
Quality of Support		X		technical	technical
Quality of Peer User Community				X	
Digital Signature		X			
Scalability		X		X	
Atomic Swap		X			
Side/Child Chains		X			
Industry Specialization		X	X		
Token Creation		X			
Security and Privacy Policies		X			
Interoperability		X		Integration	
Consensus	X	X	X		
Transaction Speed	Transaction per second	X			
Block Time		X			
Authentication		X			
Audit log		X			
Governance		X	X		GitHub Repo
Customization				X	
Ability to Understand Needs				X	
Availability of 3rd-Party Resources				X	
Timeliness of Vendor Response				X	
Activity					X
Popularity					X
Costing		Transaction		Pricing Flexibility	X

Table 1: Features proposed by websites/blogs on blockchains

Blockchain Platforms Quality Attributes			
Product		**Attractiveness**	
Attribute	**Description**	**Attribute**	**Description**
Permission	Policy for defining user roles and permissions (Permissioned, Permissionless, Both)	Governance	Organization that governs and manages the platform
Consensus	Method to assign the consensus to the miners (Proof of work, Proof of stake, Proof of importance, BFT, SFT)	Community Activity	Size and frequency of the platform developers community activity
Used Languages	Programming languages used to define a smart contract	Tool Support	Existence of tools supporting the installation and usage of the platform and/or the deployment of the smart contracts
Operative Environment	Operating systems able to host the platform	Launch Date	Date on which the first stable version of the platform has been released
Cost	Cost required to execute a smart contract on the blockchain platform	Capitalization	Value of the digitalized assets the platform manages (expressed as currencies, cryptocurrencies, precious metals, commodities, materials)
Scalability	Ease of adding resources to increase performances	Release Frequency	Frequency of release of new/updated versions
Performance	Number of transaction per Second (TPS), average time to execute a valid transaction	Documentation	Availability and quality of technical and user documentation
WaitingTime	Average time to wait for the execution of the smart contract	Business Focus	Specific business application domains
VM/Docker	Type of isolation systems used by the platform		
Tokenization	Token-based assignment policy		
Turing Completeness	Turing Completeness of the used programming languages		

Table 2: Proposed quality Framework

As far as the attractiveness is concerned, it depends on features, such as those related to the existence of technical support and adequate documentation. More features to be considered are: existence of an active community supporting the development and evolution of the platform by releasing new versions, fixing possible bugs or adding new functionalities, frequency of new releases, maturity of the platform (i.e., how old it is), business focus, its capitalization related to the transactions executed.

The proposed framework is composed of two sets of attributes, listed in Table 2. The first set groups attributes related to the perceived quality of a platform. They are indicated as ‘Product’ attributes and are reported on the left side of the Table 2. The second set of attributes characterizes the platform attractiveness. These attributes are listed in the right side of the table indicated as -‘Attractiveness’. A short description of each attribute is also provided.

It is worthwhile noticing that the proposed framework is not yet complete and definite. It represents an early solution for blockchain platforms evaluation. It can evolve as knowledge is gained and evaluation needs arise.

5 Application of the Framework

This section describes the results achieved by the application of the framework to evaluate a set of blockchain platforms, chosen for their popularity among the open source ones, to be freely used for being analysed. The section is composed of two subsections: the first lists the chosen platforms, and the subsequent one discusses the obtained results.

5.1 Chosen blockchain Platforms

In this section, a brief description of the selected blockchain platforms for smart contracts is provided, highlighting their main characteristics.

Ethereum [9] is an open source platform. It was developed by Vitalik Buterin in 2013 for Ether, the native cryptocurrency of this platform (the most important cryptocurrency after Bitcoin). It is also the most used platform for smart contracts. The platform was online on July 2015 and was successful among developers. Ethereum uses Solidity [10] as a programming language to implement smart contracts. This is an object oriented language mainly influenced by Javascript, C++, and Python. Ethereum is a public blockchain, permissionless and uses the "Proof of Work", as a consensus algorithm (however an upgrade to "Proof of Stake" is being considered for Ethereum 2.0). Each operation executed on the blockchain has a cost defined by a quantity of "gas", a measurement unit for the computation resources needed to execute the operation on a node. The "gas price" and the "gas limit" are two parameters of a transaction specifying, respectively, how many Ethers a user wants to pay for each gas unit and the maximum amount of gas the user wants to use for the transaction. Ethereum allows the creation of fungible or non-fungible tokens according to, respectively, the ERC-20 and ERC721 standards.

Hyperledger Fabric [12, 13] is one of the platforms developed in the Hyperledger project/community [11] hosted by "The Linux Foundation". The Hyperledger project/community aims to advance blockchain technologies in different application domains, such as supply chain, finance, IoT, banking, manufacturing.

Several Working Group and Special Interest Group are in Hyperledger, promoting the development of several frameworks and tools, each one with different characteristics according to the application domain and allowing a company choosing the more adequate one. Hyperledger Fabric is a modular and versatile platform that allows to adapt to the needs of different business use cases through plug and play components. Fabric has a permissioned architecture, with an open smart contract model allowing to implement different solution models, such as account model, UTXO model, structured/unstructured data. Smart contracts can be developed using general-purpose programming languages (e.g., Go, Java, Javascript, Node.js) rather than constrained Domain-Specific Languages (DSL); a support for Solidity is also provided. Fabric is not based on an own/native cryptocurrency. It provides support for pluggable consensus protocols, allowing a more effective customization to specific use cases.

Hyperledger Burrow [14] is another platform developed in the Hyperledger project/community to execute smart contracts in permissioned blockchain, as Ethereum. Contracts are usually written using Solidity, and that allows Burrow to interact with the Ethereum blockchain. The Byzantine Fault Tolerant (BFT) consensus algorithm is used in Burrow, but a Stake of Proof (SoP) algorithm can be integrated to interact with Ethereum.

Stellar [17] is a blockchain to manage currencies and payments transactions. It has its own cryptocurrency, the Lumen, but it supports any currency. Java, Javascripts and Go are the supported programming languages, while the Stellar Consensus Protocol (SCP) is used as consensus algorithm.

NEM (New Economy Movement) [15] is a blockchain developed for the XEM cryptocurrency. It was coded in Java and uses the Proof of Importance (PoI) consensus algorithm, a process similar to the Proof of Stake in which some new ranking criteria and variables/features are added to. NEM is more specifically oriented to Smart Asset management, such as currencies and supply chain. In March 2021 a new version of NEM, called *Symbol* was launched [16]. Symbol, whose cryptocurrency is XYM, is coded in C++ and uses the Proof of Stake Plus (PoS+) as consensus algorithm.

NEO [18] is an open source blockchain to support smart economy by managing digital assets and smart contracts. NEO smart contracts can be coded in several programming languages as C#, Go, Python, Java, or TypeScript. NEO provides many features similar to the ones from Ethereum. Indeed, NEO base asset is the NEO token that is used to generate "gas" tokens by which to pay transactions fees to be executed on the blockchain. It allows cross-chain interoperability with Ethereum and more other platforms. The consensus mechanism in NEO is the delegated Byzantine Fault Tolerance (dBFT), based on the PBFT (Practical Byzantine Fault Tolerance) algorithm.

5.2 Analysis of the chosen blockchain platforms

The listed platforms have been used in a case study for verifying the applicability and usefulness of the framework. Some graduated students were assigned to use the framework to evaluate the platforms to select the one more suitable for smart contracts in an e-Commerce application, satisfying specified requirements. At this aim the platforms were installed, their available documentation was analyzed, and they were used to develop and execute some smart contracts. Each attribute considered in the Framework was assigned a value, allowing to perform a comparison among the platforms. Table 3 reports the results of such an evaluation; the rows of the Table report the Framework's attributes, while each column is assigned to one of the analyzed platforms.
The first quality attribute to be considered in the perceived quality of the product is *Permission*. It allows the definition of the responsibilities of the technology users with reference to the platform by establishing if it is Permissionless or Permissioned. Table 3 shows that just Ethereum does not include any permission system, while the NEO and Hyperledger platforms are Permissioned; Stellar and NEM platforms allow both types of Permission.

Then, the mechanisms used for establishing the Consensus are analysed. For this feature, the different platforms apply different rules. For example, the Hyperledger Fabric platform adopts a democratic system and applies the Byzantine fault tolerance (BFT) mechanism, on the basis of which the consensus derives from the qualified majority (50% +1) of the miners. A consensus mechanism oriented towards the Proof of Work is applied, for which the miners, for gaining consensus, have also to solve a complex puzzle (a very computationally complex mathematical function) strongly affecting the performance of the consensus and then the scalability and performance. The Consensus Protocol (SCP) of Stellar platforms divides the group of miners into slices, each of which can choose a custom rule for validation. In this way, the node validating the transaction delegates to the underlying network to choose the best way to define the truthfulness of the calculated result.

Solidity is the main language used in Ethereum. The Solidity language is also used in Hyperledger Burrow, allowing it to join the advantage of a Permissioned platform and the Ethereum world. Solidity is designed to target the Ethereum Virtual Machine (EVM) and this guarantees the isolation in Ethereum, as well as in Hyperledger Burrow. Table 3 also shows that the large part of the considered platforms uses Java as language for both their implementation and for their services, then, a (Java) developer who approaches these platforms does not have to learn a new programming language.

Just the Hyperledger Fabric platform has no Cost to execute smart contracts, while all the other considered platforms exhibit costs for either deployment or execution of smart contracts.

The Scalability attribute impacts on the platform success, and this represents an obstacle for Ethereum with a low scalability, in fact the developers are searching new solutions. On the other hand, the characteristics of performance, scalability and a modular type of consensus allow Hyperledger Fabric to better respond to the needs of reactivity.

The larger diffusion of Ethereum and Hyperledger motivates their higher Capitalization. Thanks to the support that Ethereum and Hyperledger offer to the developers, their values for Community Activity is high. In Ethereum this has led to the implementation of many solutions that can represent the basis for implementing new ones. In addition, in Ethereum is defined a set of standards to guarantee the quality of the Solidity code, that, together with the realized environment for supporting the community, has had a great impact on the community. Analogously, Linux Foundation and IBM, which have created Hyperledger, besides the continuous development made available tools to guide and support developers both in both the design and in the release phases.

Platforms	**Ethereum**	**Hyperledger Fabric**	**Hyperledger Burrow**	**Nem**	**Stellar**	**Neo**
Product						
Permission	Permissionless	Permissioned	Permissioned	Both	Both	Permissioned
Consensus	Proof of Work	BFT	BFT	Proof of Importance	SCP	BFT
Used Languages	Solidity, LLL, Serpent	Go, Node.js, Java	Solidity	Java	JavaScript, Java, Go	Javascript, Java, Go, Python
Operative Environment	No	Yes	No	Yes	Yes	Yes
Cost	Yes	Free	Yes	Yes	Yes	Yes
Scalability	No	Yes	Yes	Yes	Yes	Yes
Performance	No	Yes	Yes	Yes	Yes	Yes
Waiting Time	Yes	No	No	No	No	No
VM/Docker	VM	Docker	VM	Docker	Docker	VM
Tokenization	Yes	No	Yes	Yes	Yes	Yes
Turing Completeness	Yes	Yes	Yes	Yes	No	Yes
Attractiveness						
Governance	Ethereum	Linux	Linux	Nem	Stellar	NEO
Community Activity	Yes	Yes	Yes	Low	No	Yes
Support Tools	Yes	Yes	Yes	No	No	Yes
Launch Date	Jul 2015	Mar 2017	Apr 2017	Mar 2015	Jul 2014	Jun 2015
Capitalization	High	High	High	Low	Low	Low
Release Frequency	High	High	High	High	High	High
Documentation	Detailed	Detailed.	Detailed	Detailed	Detailed	Detailed
Business Focus	Cross	Cross	Cross	Cross	Finance	Smart Economy

Table 3: Results of the Evaluation

Concerning Turing Complete attribute, the different platforms support it, with the exception of the Stellar platform.

Once the blockchain platforms have been evaluated, the collected values can be used for choosing the most suitable one, on the basis of specific implementation needs. For example, if the software engineer wants to use an open platform that is well supported, the choice can be Ethereum as it is Permissionless, and the value of Capitalization is high. In fact, for this platform, the documentation and the available guides, referred to the Documentation and Community attributes, encourage the developer to use it. Also Stellar and NEM would be considered for adoption being Permissionless, but they have a lower Activity Community. In addition, the choice cannot be Stellar because it does not support Turing Completeness. Hyperledger Fabric and Burrow are not be suitable because they are Permissioned.

6 Conclusions

This paper considers the blockchain platforms and discusses their main characteristics. In the last year, many technological solutions have been proposed with reference to blockchains. However, aspects concerning the definition of standards and quality evaluation have not been completely analysed. The existing approaches mainly consider performance and security features. Then, on the basis of the performed analysis, the paper has defined a preliminary evaluation framework whose aim is to provide support to a software engineer that wants to choose one of the existing blockchain platforms for implementing and executing smart contracts. In particular, the framework offers support for defining the external quality of the blockchain platforms, as perceived by a possible user with reference to her/his need, and the platform attractiveness.

The proposed framework has been applied, in a case study, to analyze and evaluate a set of open source blockchain platforms, appearing among the most popular ones. Their analysis and evaluation showed that a platform that is the best in absolute does not exist, but each platform is valid with reference to specific needs. The case study showed the usefulness and the applicability of the proposed framework. However, more experiments are needed to better confirm this result, as well as its validity.

The proposed framework is a preliminary one and it can be improved by considering more quality attributes for capturing further aspects regarding blockchain platforms and smart contracts. Finally, further effort has to be spent for better refining the quality attributes to allow for a more objectively and widespread assessment.

References

[1] Szabo, N. “Smart Contracts,” 1994, http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

[2] K. Christidis, M. Devetsikiotis, “Blockchains and Smart Contracts for the Internet of Things” IEEE Access 4(1), pp. 2292 – 2303, 2016.

[3] F. Daniel, L. Guida, “A Service-Oriented Perspective on Blockchain Smart Contracts” IEEE Internet Computing 23(1), pp. 46 – 53, 2019.

[4] S. Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence. AI Matters, 1(2):19–21, Dec. 2014.

[5] Dylan Yaga, Peter Mell, Nik Roby, Karen Scarfone, “NIST IR 8202 - Blockchain Technology Overview”, National Institute of Standards and Technology, Oct. 2018, https://doi.org/10.6028/NIST.IR.8202

[6] S. N.Khan, F. Loukil, C. Ghedira-Guegan,E. Benkhelifa, A. Bani-Hani "Blockchain smart contracts: Applications, challenges, and future trends", Special Issue on Blockchain for Peer-to-Peer Computing, on-line Apr. 17, 2021, https://doi.org/10.1007/s12083-021-01127-0

[7] John M. Medellin and Mitchell A. Thornton, "Consideration of Quality Attribute Tradeoffs of the Blockchain Pattern in the Software Development Process", Annals of Emerging Technologies in Computing (AETiC), Vol. 3, No. 4, 2019

[8] B. Koteska, E. Karafiloski, A. Mishev, "Blockchain Implementation Quality Challenges: A Literature Review", Proceedings of the SQAMIA 2017: 6thWorkshop of Software Quality, Analysis, Monitoring, Improvement, and Applications, Belgrade, Serbia, 11-13.9.2017, Also published online by CEUR Workshop Proceedings (http://ceur-ws.org, ISSN 1613-0073)

[9] Vitalik Buterin "Ethereum: a next generation smart contract and decentralized application platform", white paper, https://ethereum.org/en/whitepaper/, 2017

[10] "Solidity 0.8.6 documentation", https://docs.soliditylang.org/en/v0.8.6/

[11] "Solidity" https://en.wikipedia.org/wiki/Solidity]

[12] https://www.hyperledger.org/

[13] https://www.hyperledger.org/use/fabric

[14] hyperledger_fabric_whitepaper.pdf, at https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger_fabric_whitepaper.pdf

[15] https://www.hyperledger.org/use/hyperledger-burrow

[16] https://nem.io/platforms/

[17] https://docs.symbolplatform.com

[18] https://www.stellar.org/

[19] https://neo.org/

[20] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system, https://bitcoin.org/bitcoin.pdf, 2008

[21] B. Hu, Z. Zhang, J. Liu, Y. Liu, J. Yin, R. Lu, X. Lin, A comprehensive survey on smart contract construction and execution: paradigms, tools, and systems, Patterns, 2(2), 2021, (https://www.sciencedirect.com/science/article/pii/S2666389920302439)

[22] F. Casino, T. K. Dasaklis, C. Patsakis. A systematic literature review of blockchain-based applications: current status, classification and open issues. Telematics and Informatics, Vol. 36, pp. 55–81. March 2019. doi: 10.1016/j.tele.2018.11.006

[23] https://www.itransition.com/blog/smart-contract-platforms

[24] https://www.g2.com/categories/blockchain-platforms

[25] https://www.leewayhertz.com/blockchain-platforms-for-top-blockchain-companies

[26] https://www.gartner.com/reviews/market/blockchain-platforms

[27] https://hackernoon.com/top-blockchain-platforms-to-watch-out-in-2019-aa80e336a426

[28] S. Bouraga, A taxonomy of blockchain consensus protocols: A survey and classification framework. Expert Systems with Applications. Vol. 168, Elsevier, 2021