Proceedings of the 53rd Hawaii International Conference on System Sciences | 2020

Applying Software Quality Criteria to Blockchain Applications: A Criteria Catalog

Hauke Precht
Carl von Ossietzky University
Ammerländer Heerstr. 114-118
26129 Oldenburg
hauke.precht@uol.de

Stefan Wunderlich
Carl von Ossietzky University
Ammerländer Heerstr. 114-118
26129 Oldenburg
stefan.wunderlich@uol.de

Jorge Marx Gómez
Carl von Ossietzky University
Ammerländer Heerstr. 114-118
26129 Oldenburg
jorge.marx.gomez@uol.de

Abstract

The selection of the suitable blockchain software ecosystem has become very complex, given the growing market. More and more products with different functionality (mainly consensus algorithms and smart contracts) are available on the market. To identify the correct blockchain system for the respective application, a catalog of criteria with a focus on software quality is developed in this work. This catalog supports the selection of the right application and can be individually weighted.

1. Introduction

There are numerous blockchain applications and approaches for several specific domains as well as a set of applications aiming for general usage in heterogeneous use cases. When starting to get familiar with blockchain and its possible usage, the actual need for a blockchain must be determined as the first step. Various studies have already addressed the question of whether a blockchain is useful as a software solution for the respective use case or not.

In order to determine whether a blockchain can be used sensibly or whether classical relational databases are desirable the works of [1] and [2] can be applied. These works provide clear guidelines on whether the respective use case calls for a blockchain implementation or not. Combined with the approach of a taxonomy for blockchains proposed by [3, p. 252], it is possible to identify which type of blockchain can be applied to solve a specific problem. The next step would be to investigate existing blockchain applications. A simple review for the number of existing cryptocurrency implementations (which are mostly based on blockchain technology) reveals many possibilities (there are 2140 cryptocurrencies listed on CoinMarketCap [4]). Most cryptocurrencies are based on public blockchains. But as companies started to adopt the technology as well, also private, and permissioned blockchains emerged. While public blockchains often implement proof of work consensus (e.g., Bitcoin, Ethereum, Litecoin, etc.), private and permissioned blockchains (e.g., R3 Corda, Hyperledger Fabric, etc.) in most cases implement completely different consensus algorithms.

In the area of enterprise blockchain systems, a large set of applications exist. However, all these applications are in different stages of development and are not suitable for every use case. For many companies, it is a big challenge to identify the right blockchain technology for the respective use case.

This paper shows an approach based on a criteria catalog in order to help companies to choose the correct blockchain implementation. The criteria catalog is based on well-known software evaluation criteria, such as ISO 25010, capability maturity model (CMM), and quality of open source software (QualOSS). This paper is structured as follows: First, the used methodology and the related background is presented. Next, the identified criteria are introduced. This section is split into four subsections covering blockchain-specific criteria, software quality criteria, open-source software quality criteria, and software maturity models. Subsequently, a summary of the identified and selected criteria, alongside with an example application, is given. This paper concludes discussing the application of methods for multi-criteria decision analysis (MCDA) using the criteria catalog.

2. Methodology and Background

This work has been developed using grounded theory and literature review methods. In order to determine the various criteria, a literature review was first carried out in order to capture the essential aspects within the scope of the software quality criteria. These works were then prioritized. The prioritization was

URI: https://hdl.handle.net/10125/64511
978-0-9981331-3-3

carried out based on the relevance of the underlying works so that only those works for the criteria catalog were selected that were highly accepted in the scientific community. Furthermore, attention was paid to ensuring that the works were up to date so that only the most up-to-date approaches were integrated. In the field of cryptocurrency and distributed ledger/blockchain technology in general, numerous works addressing classification have already been published. However, all these approaches have in common that they single out certain partial aspects, but do not provide a holistic picture of the technology. In this section, the existing works, extracted from IEEE Xplore, ACM Digital Library, and ScienceDirect, are briefly presented to give an overview of existing approaches.

An early taxonomy for distributed consensus algorithms, focusing on cryptocurrency systems was proposed by Glaser and Bezzenberger in 2015. The purpose of this taxonomy is to enable practitioners and researchers to classify a new cryptocurrency implementation (or one which is being developed) into the existing systems landscape of cryptocurrency implementations [5].

In their article, from 2016, concerning blockchain technology and smart contracts for the "internet of things," Christidis and Devetsikiotis refer to a taxonomy based on questionnaire approach that covers the access to the network, access to transaction permission, and mining permission. Furthermore, they propose to evaluate the used transaction mode, i.e., the unspent transaction output model or the account-based model, which allows the usage of smart contracts [6].

Based on the different access levels, a differentiation between public, private, permissioned, and permission-less blockchains can be made. Such differentiation is used to populate heterogeneous decision trees providing guidance in the process of selecting a blockchain implementation for a specific use case.

Peck et al. proposed one such decision tree [1]. Opposed to public opinion, [1] identified that "it is rather difficult to identify a useful application for blockchain." Questions about the underlying use case, determine, step by step if blockchain is a desirable technology. Furthermore, the decision tree tries to identify the access needs, such as the access levels proposed in [6]. If data must be kept private, a permissioned blockchain should be considered. If it is data that can be publicly accessible, a public blockchain is a possible solution [1].

A similar approach is described by [2] for the National Institute of Standards and Technology (NIST). Their decision tree starts with a similar question-based system. Firstly, they ask if a shared data storage is needed. Secondly, they ask if multiple entities can provide data. It must be reviewed if this data is private and if it requires to be immutable. Lastly, it is verified if the data must be tamper-proof. If all these questions are answered positively, Yaga et al. conclude that it might be a useful blockchain use case.

Wust and Gervais propose a similar decision tree, which may lead to four different results, targeting the already described access levels: permissionless blockchain, public permissioned blockchain, private permissioned blockchain or the recommendation to not use a blockchain in the first place [7]. Aspects that are considered in their work include sorting of states, the existence of multiple writers, usage of trusted third parties, known and trusted writers, and public verifiability [7].

A similar approach is described in [8], where experts identified five key questions that should be answered in order to determine if a blockchain should or could be used. First, they ask if a shared database is required. Next, it should be identified if multiple parties require write permissions. Thirdly, they ask if these identified parties are potentially untrusted. The fourth identified question targets the need for disintermediation. The last question aims to identify if it is necessary to see the links between transactions [8].

Xu et al. propose a taxonomy, using basic questions as a starting point but drill down into further detail. Their taxonomy provides an overview of blockchain-specific architectural aspects and their impact on design decisions. [3]. These aspects, along with the possible impact, are discussed in three tables. This taxonomy is intended to aid software architects "to evaluate and compare blockchains" [3]. The main point of criticism is the insufficient explanation of the impact of different properties as well as the selection of said properties. Wessling et al. classify the work of Xu et al. as very specific and for a single blockchain system, focusing on blockchain-specific technical details, such as consensus algorithms [9]. Wessling et al. provide an approach "to decide which elements of an application architecture could benefit from the use of blockchain technology" [9], concentrating on the embedding of blockchain in existing software environments.

The approaches and taxonomy described above are intended to be independent of use case. In opposite to this, Fridgen et al. developed a framework, based on an evolutionary approach, specifically for the public domain [10]. Within this framework, they identified three domains: the technical, functional, and legal domains. Their focus is to "derive a conceptual framework that unifies blockchain concepts and their relationships to digital market models into a single framework." [10].

Another use case-specific, taxonomy was introduced by [11] concerning the post-trade process

within the financial sector. They developed a method for creating requirement-driven taxonomies and evaluated this method in the mentioned post-trade use case. They take domain-specific requirements from the technological, socio-economic, and legal environment into consideration and add blockchain-specific attributes. These blockchain-specific attributes are mainly derived from [3]. The review of related work shows that most works are dealing with the general question for the applicability of blockchains along with taxonomies evaluating blockchain-specific attributes. These approaches can be used for decision-making on a strategic level. Once the decision is made in favor of a blockchain, the question arises, which kind of blockchain should be used, and if it might need to be developed from scratch. Every approach found so far is missing a guideline in terms of comparing and selecting blockchain technology after identifying the need for one in dependence of the use case. The identified works especially lack the consideration of technical quality criteria. The aim of this work is to fill this gap by taking up existing works and developing a catalog of criteria on this basis.

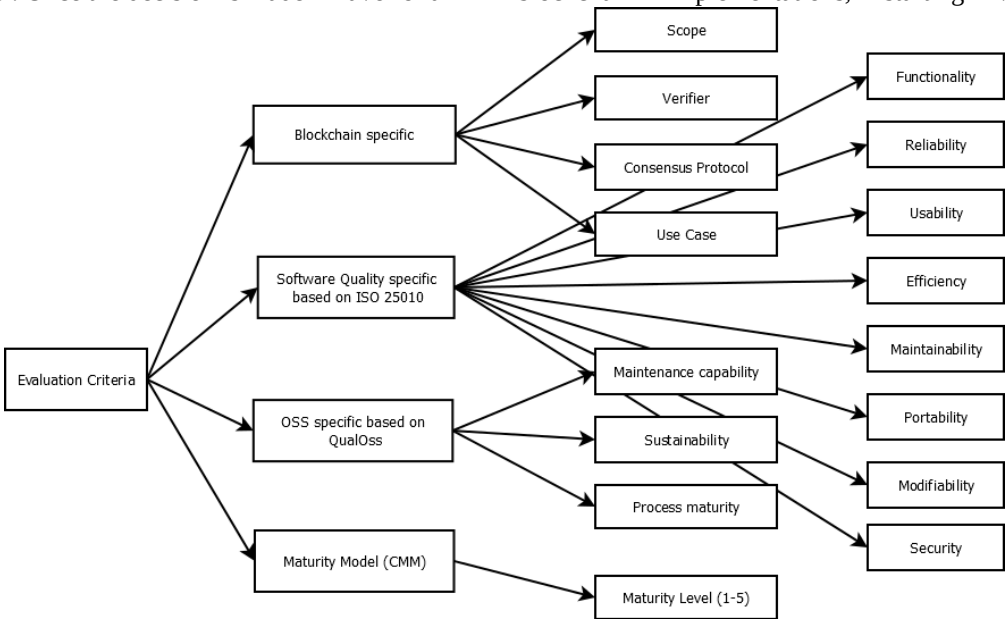

Figure 1. Criteria overview and origin

3. Overview

As described in the section above, current works deal with blockchain-specific criteria only. They do not consider that; besides these specific criteria, further general criteria should be considered to select a blockchain implementation. Software quality criteria, for example, ISO 9216 and its successor ISO 25010 [16], can be considered. Moreover, software quality models emerged which focus on the maturity of the development processes and the respective organization that provides the software. With the rise of open-source software (OSS), new quality models were developed to meet OSS-specific criteria, such as the community. Four points of view: blockchain specific, software quality, OSS quality, and software maturity, have been identified as important when it comes to selecting a blockchain implementation. In the following, actual model implementations are introduced, and it is discussed if they could be of use when evaluating blockchain implementations, starting with the blockchain specific point of view.

The taxonomy proposed by [3] is used to cover the **blockchain-specific criteria**, as it covers major blockchain aspects and is commonly used cited by many authors. As every blockchain implementation is a software, that needs to be deployed, maintained, and extended by a set of software developers, **software-specific quality criteria** must be considered as well.

Within the literature concerning software quality models, a range of different approaches exist. Five well-known quality models (McCall's Quality Model [12], Boehm's Quality Model [13], Dromey's Quality Model [14] and FURPS Quality Model [15] and ISO 9216 (succeeded by ISO 25010 [16])) were analyzed and compared by [17]. As they conclude in their research, most of the quality models focus on one perspective, e.g., the product perspective. Only ISO 9216 offers a comprehensive view as well as the top-down, and bottom-up approach [17]. Most of the described models use similar criteria or the full subset of ISO 9126. The

ISO 9126 standard is succeeded by ISO 25010 which is why it is used in this work to identify relevant software quality criteria.

As most blockchain implementations are open source, the third point of view will be **open-source specific criteria**. The unique nature of open-source software (OSS) requires unique quality evaluation criteria [18], [19]. There are several OSS quality models, such as QSOS [20] or QualOSS [21]. Every one of them considers two major quality perspectives: the product perspective, and the community perspective. As [18] point out, the community is a unique attribute of OSS and can be considered the main difference in opposition to commercial software [22]. Only QualOSS also considers a process perspective within open-source software [22].

Since blockchains are used for potentially critical business areas (e.g., finance, healthcare, governmental), it is crucial to select a blockchain that is mature enough from a process point of view [23]. Therefore, **software maturity criteria** are used to evaluate maturity from a process perspective. Since blockchain is a new trend and multiple blockchain implementations are still emerging, problems resulting from insufficient maturity may occur. As already described, QualOSS considers the process maturity with the focus on community-driven processes. Software maturity models evaluating a larger range of process maturity and will be therefore integrated as an aspect on its own. Several software maturity models emerged within the last 20-30 years. The literature review by [24] shows that 58% of maturity models examined are based on the capability maturity model (CMM) [23]. Another study conducted by [34] shows that 60% of the models evaluated are based on CMM. Based on the high percentage of before conducted research, CMM will also be used in this work. It is evaluated in terms of applicability for blockchain software.

As shown in Fig. 1, an attempt is made to combine the four identified points of view – *blockchain specific*, *software quality*, *open-source software quality* and *software maturity* – towards a general applicable criteria catalog to evaluate blockchains. Subsequently, the four different domains with their specific evaluation criteria are described in greater detail.

3.1. Blockchain-specific Criteria

Within this section, criteria which reflect the specific characteristics of blockchains are considered. Naturally, these specific criteria need to be considered when aiming to create a general approach for evaluating and comparing blockchains. As already mentioned, [3] provides a taxonomy based on a large set of well-established blockchain specific attributes and will, therefore, serve as a basis for identifying relevant criteria. In total, four criteria are selected:

Scope: The scope of the blockchain describes the accessibility of the blockchain for the participants. Blockchains are classified as public or private and permissioned or permissionless, respectively [25], [26], [27]. If everyone can participate in a blockchain, it is considered public. If only a restricted set of participants have access to the blockchain, it is called private as different use cases require different accessibility, the scope of the blockchain needs to be determined.

Verifier: Xu et al. point out that there are different possibilities of how blocks or transactions are verified. It is possible that a single verifier exists, trusted by the whole network. The second possibility is an M-of-N verifier who vote which proposed block is appended to the blockchain. The third option they identified is the ad hoc verifier [3]. Depending on the characteristics of the verifier, the need for a consensus protocol might differ.

Consensus protocol: Blockchain systems use distributed consensus algorithms to agree on the order of how elements are appended to the chain. They also provide a continuous service [28]. That means they are a key element of every blockchain. Depending on the blockchain scope, the consensus protocol varies. Zheng et al. point out, that private blockchains might favor practical byzantine fault tolerance (PBFT) [29], while in public blockchains, typically proof of work (PoW) or proof of stake (PoS) algorithms are used. Depending on the scope of the blockchain and the possible splitting of permissions (who can mine new blocks), supported or used consensus protocols within the blockchains need to be evaluated.

Use case: Different blockchains were developed for a specific domain, often for financial technology. Depending on the use case and its domain, this needs to be considered when selecting a blockchain. Ethereum, for example, “attempts to build the generalized technology; technology on which all transaction-based state machine concepts may be built.” [30]. Hyperledger Indy, in opposite, focuses on the specific domain of decentralized identity [31]. That means that the use case must be considered when evaluating blockchains.

Blockchain-specific criteria depend on each other to a certain degree, e.g., a private blockchain might use a single verifier with no need for a consensus protocol. A blockchain used for creating a cryptocurrency most likely will be a public blockchain, requiring a proof of work or proof of stake consensus protocol.

3.2. Software Quality Criteria

As already described, a blockchain is a piece of software that needs to be maintained, deployed, and extended by a set of developers. When introducing a

blockchain (or any new software system) into a corporate environment, it must be determined if this software meets specific quality criteria. If, for example, the technical environment consists of large heterogeneous systems, portability would be a factor to be considered. Besides, there is a set of quality criteria which software should fulfill to be useable from a technical point of view. In the following, the main aspects of ISO 25010 are introduced, and it is shown how they can be applied to evaluating blockchains.

Functionality suitability: This is the first aspect of ISO 25010. It is used to check if the software provides the required functionality [16]. As different domains have different functional requirements, which need to be provided by the blockchain application or which need to be developed on top of the blockchain, this factor is taken into consideration. Next, to the actual functionality, sub-factors like compliance or security are also included. The latter is especially important when dealing with important transactions.

Reliability: This factor describes how reliable software is in terms of fail-safety. Sub-factors are, for example, the fault-tolerance or the maturity of the software. Since blockchains are decentralized, the fault-tolerance is an important factor also in terms of malicious attacks. Due to the fast-evolving blockchain technology, the maturity should be taken into consideration as well, to determine possible outcomes of future developments.

Usability: Usability can be seen from multiple perspectives, e.g., from a developer's or a user's point of view. As blockchain is a low-level software that does not directly affect the user interface (UI), the end-user point of view can be neglected. The developer's point of view, however, should be considered in terms of learnability or understandability. As every software must be maintained by a set of (sometimes fluctuating) developers, understandability is a critical feature every software should provide. Especially when the software will be used, which was not developed in-house, this criterion should be focused as there will be the need to add and modify or at least deploy the software.

Performance Efficiency: This quality factor determines how efficient the software works. Further subfactors are time behavior, resource utilization, and capacity [16]. Since the blockchain runs on multiple, heterogeneous systems with different hard- and software specifications, especially the resource behavior, should be evaluated.

Maintainability: As the blockchain needs to be further enhanced by different developers, the requirement for maintainability is an important one. Maintainability is within the ISO 25010 further split into modularity, reusability, analyzability, modifiability (combining changeability and stability from ISO 9126) and testability [16]. The analyzability, as well as modifiability, will heavily affect the quality of future developments and should be well investigated.

Portability: Portability describes in what way a software can be ported to another environment. A sub-factor is the adaptability that describes how the software reacts to changes within its environment. As mentioned above, the blockchain needs to support a set of heterogeneous systems, making the portability criterion necessary.

Modifiability: The modifiability is not a "top-level" factor in the ISO standard but a sub-factor of the maintainability [16]. In Boehm's model, however, it is covered at a higher level labeled as modifiability [13]. Already in 1976, they identified that it is crucial to evaluate how efficient it is to maintain or to modify a newly acquired software. In order to be of use for specific domains, blockchains must implement domain-specific requirements. Especially general-purpose blockchain application approaches like the Hyperledger project need to be customized for the respective domain. Therefore, this criterion is, in this work, on a higher level than it is currently in the ISO standard.

Security: This factor describes the "[…] degree to which a product or system protects information and data […]" [16]. Only users or software systems with appropriate authorization should access the data or information they need. Within security, five sub-factors exist: confidentiality, integrity, non-repudiation, accountability, authenticity. As blockchains are used to store and manage several types of data and information, e.g., transaction data, security must be provided and needs to be considered for evaluating blockchains. Due to the decentralized approach combined with cryptographically secure linkage of blocks; the sub-factor integrity should be fulfilled by nearly every blockchain in terms of preventing unauthorized modification.

3.3. Open Source Software Quality Criteria

As shown the "classic" software quality models focus on the software only. With an increasing number of open source software projects, the conventional software quality models were not sufficient anymore as they do not consider the community of a software or the process maturity. Therefore, open source software quality models were introduced. The identified starting point concerning open source software quality criteria is, as already mentioned, QualOSS. The product quality perspective is similar to the above-mentioned software quality criteria as they use the same standards, i.e. ISO 25010 [32] [22]. As these criteria were already investigated, they will not be considered within this section again, but the community perspective will be

further analyzed in terms of applicability for blockchain projects. The community consists of developers and users which contribute to the software. The community criteria can be further split up in several sub-criteria concerning the maintenance capacity, the sustainability and the process maturity [22] which will be discussed in the following and mapped to the use case of blockchain evaluation.

Maintenance capability: The maintenance capability covers the essential questions if the community can maintain the software throughout a longer period and if they follow established processes to secure a certain degree of quality. Within the QualOSS model, the analysis of existing mailing lists, forums and ticket systems can be used to analyze the maintenance ability of the community [21]. Based on this data, it is possible to identify the core contributors of the software. Most open source blockchain projects are hosted on GitHub. Several studies within the field of social analytics and social coding analyzed GitHub projects and its developer base, for example in terms of relations between GitHub users and repositories as well as their expertise [33]. As most blockchain implementations are open source software and are hosted on GitHub, the community of these projects need to be regarded when evaluating an open source blockchain implementation.

Sustainability: This criterion describes the ability of the community to sustain and to remain in order to maintain and develop the software [21]. This means that the sustainability is strongly connected with the maintenance capability. Therefore, it is considered as well when evaluating blockchain implementations. Possible metrics that are considered in order to measure the sustainability would be the rate of developer intake, turnover, or the overall growth in terms of active developers. These metrics are grouped as the factor "developer base" in the Software Quality Observatory for Open Source Software model (SQO-OSS) [19].

Process maturity: This criterion describes how mature the software is, i.e. how well established the processes within the community are. These processes describe how a new feature is introduced or in what way a bug is fixed. Since companies must rely on the community to introduce features and bug fixes to a certain degree, the process maturity needs to be considered especially in crucial blockchain projects. In order to evaluate the process maturity, several factors can be used, for example, if a project management structure can be determined or if a quality assurance process is established. These criteria are also part of the Qualification and Selection of Open Source (QSOS) Model [20].

3.4. Software Maturity Models

The established processes within open-source software projects are regarded as significant contributing factors. These process maturity criteria from the open-source software community can be directly linked to software maturity in general. The CMM will serve as the basis for this section. It is determined whether it can assist in evaluating blockchains from a process point of view extending the process maturity factor described above.

CMM provides five levels to describe the maturity of a software: initial (level 1), repeatable (level 2), defined (level 3), managed (level 4) and optimizing (level 5), where initial is the lowest level and optimizing is the highest reachable level [23]. For each level, a set of characteristics is defined by *Paulk et al.*, which must be met, in order to reach the next level. In the following sub-sections, the different levels, along with the goals which must be fulfilled in order to reach that level, are described based on [23]:

Initial (level 1): The initial level does not have any criteria to be met, i.e., every software is at least at this level [23]. If a (blockchain) software is identified to be at the initial level, it hints that no process of software management is established.

Repeatable (level 2): If the software process includes requirements management, software project planning, software project tracking and oversight, software subcontract management, software quality assurance, and software configuration management it can be considered as repeatable and is therefore on level 2 [23]. Fulfilment of these requirements is evaluated by checking if known project management tools, such as Jira, Tempo or Confluence, are used.

Defined (level 3): A software process can be defined (in level 3) in case organization process focus, organization process definition, training program, integrated software management, software product engineering, integrated group coordination and peer reviews are in place [23].

Managed (level 4): Level 4, managed, is reached when a quantitative process management, along with a software quality management is introduced to the software process [23].

Optimizing (level 5): The highest level, optimizing, is reached once a defect prevention, a technology change management, and a process change management is in place [23].

As shown, each level represents an optimization of processes concerning the software. When introducing a blockchain, this is a crucial part as it can be derived by the maturity level how robust the software is as well as how the software is supported. Therefore, it is included in the criteria catalog.

4. Summary of Criteria

In the above sections, a selection of different criteria is presented and discussed if it is feasible to include these into a selection process for blockchain implementations. Below, these factors are summarized with a short description. There are several possible ways to measure and to identify possible information to meet a single criterion. As the process maturity criterion can be considered alike, the maturity levels identified when considering software maturity models. They are merged into one criterion.

Scope: It must be analyzed if the blockchains are private, public, permissioned, or permission-less. This information can be obtained from respective whitepapers or technical analysis

Verifier: The number of verifiers approving transactions must be determined. This depends on the scope, and the information can be gathered from the whitepaper as well.

Consensus protocol: The consensus protocols are strongly linked to the number of verifiers, e.g., a single verifier does not need to find a consensus. For evaluating which consensus protocols are supported, the respective whitepapers can be used as well as reviewing the source code directly.

Use case: It must be considered if a blockchain was developed for a special use case or if it should provide a basis for multi-purpose applications. Again, this information can be obtained from whitepaper and from the company's website.

Functionality: The technical functionality of the to be evaluated blockchain implementation must match the requirements of the use case.

Reliability: Depending on the use case, the blockchain software must serve as a reliable source of data and therefore, must fulfill this quality criterion.

Usability: The usability from a developer's point of view must be taken into consideration in terms of modifiability and available documentation. The end-user perspective can be neglected as they do not interact with blockchain directly.

Efficiency: Efficiency has, for example, to be considered in terms of transaction throughput depending on the expected usage and the application domain. Statistics concerning the efficiency can be obtained from existing studies or by conducting proof of concepts and own measurements.

Maintainability: Blockchain software must be maintained by a set of developers. Maintainability can be derived from several factors like testability (can be measured by the number of existing unit tests) or stability (can be measured by examining reports of the continuous integration tools)

Portability: Blockchain software should be easy to install and should support multiple environments so that it can be run by multiple heterogeneous parties building the network. Portability can be tested by evaluating the necessary installation steps and existing scripts.

Modifiability: The modifiability must be evaluated in order to determine if the application can be modified to fit the exact requirements of the use case.

Security: Especially in private blockchain implementations, it must be evaluated if access rights are integrated.

Maintenance capability: The community must show that they can maintain the core blockchain implementation and provide updates as these are necessary when building a software stack.

Sustainability: Sustainability describes the likelihood of the community to sustain and to further develop and maintain the blockchain implementation. This is strongly linked to the before-described maintenance capability.

Process maturity: The process maturity provides an insight into how well the community is established and considers processes regarding the integration of new features, bug fixing, or release management.

Maturity level: The level of maturity of a blockchain application based on CMM indicates how well the community or company providing the blockchain implementation is organized from a broad process point of view.

Criteria from different aspects may have an impact on other criteria. The process maturity from the open-source software community quality criteria can affect the functionality criterion stated in ISO 25010. This could be, for example, the case when no working quality assurance process is defined, which leads to a higher possibility that software errors are not found. Further possible implications could be the used consensus protocol and the reliability criterion. The above-described criteria will be briefly applied in an example in the next section.

5. Example

A brief, exemplary application of the criteria catalog is shown in table 1. The identified criteria are applied to the three most widely used [35] blockchain implementations: Bitcoin, Hyperledger Fabric, and Ethereum. Since the importance of the criteria may differ between use cases, no weighting of the criteria is done in this work. However, to give a rough idea of how the use case could influence the weighting of the criteria, consider the use case identified by [44]. The authors describe a blockchain-based system to digitize the bills of lading leveraging blockchain technology. In

this use case, only a considerable small set of actors should be able to access data, leading to a higher weight of the *scope* attribute.

The example in table 1 shows that it can be distinguished between qualitative and quantitative criteria. Qualitative criterions do not leave any room for interpretation, for example: Bitcoin is a public blockchain and it uses a PoW consensus protocol. Another example is the criterion *efficiency*, which depends on measurable metrics, such as the transactions per second (TPS).

Table 1: Applying the criteria catalog

	Bitcoin	**Hyper-ledger**	**Ethereum**
Scope	Public [36]	Private [37]	Public
Number of Verifiers	~ 9962 [36]	Configurable [37]	~ 8829 [38]
Consensus Protocol	PoW [36]	Kafka / Raft [37]	PoW [39]
Use case	Crypto Currency [36]	Multi-purpose [37]	Crypto Currency / multi-purpose [40]
Functionality	-	-	-
Reliability	5	4	5
Usability	4	4	5
Efficiency	1 (4.6 TPS [40])	5 (20000 TPS [41])	3 (15 TPS [42])
Maintain-ability	2	5	3
Portability	4 [43]	3 [41]	4
Modifiability	2	5	3
Security	3	5	3
Maintenance capability	5	4	4
Sustainability	5	4	4
Maturity level	4	4	4

The data for quantitative criteria must be obtained through research, prototyping, and own expertise. For example, a public blockchain is out of the question for an application for electronic bills of lading, since the necessary confidentiality is not given. Furthermore, there may be other factors that are decisive, e.g., a blockchain framework may not support smart contracts. This may mean that the necessary transactions cannot be mapped. This applies to different use cases and must, therefore, always be individually determined. In this example, an exploratory approach is used where each blockchain framework was investigated individually.

In this example, a scoring from 1 to 5 is used to evaluate the criteria, where 1 is the worst, and 5 is the best possible. The criterion for functionality is not considered in this example because it requires an in-depth analysis of a specific use case.

In this example, all criteria and their characteristics have been depicted on a nominal scale. This is intended to illustrate how the criteria catalog can be applied. However, the focus of this work is on identifying the criteria. Future work will show more comprehensive application examples.

6. Conclusion

This paper provides a compilation of different criteria to select blockchain implementations for different use cases. Current approaches concerning the selection of blockchain implementations solely focus on the applicability of blockchain technology. As shown in the background section, taxonomies were developed based on blockchain-specific attributes, such as the access scope or the used consensus protocols. All the identified approaches do not consider software quality, open-source software quality, or software maturity models.

This paper shows, that these quality criteria, combined with blockchain-specific criteria, lead to a general criteria catalog, enabling practitioners and researches an in-depth evaluation of blockchain implementations and their applicability in specific use cases. The presented catalog is based on well-established models and approaches. The criteria for software quality are extracted from the ISO 25010 standard (formerly ISO 9126). In order to evaluate open-source software quality, the QSOS model is integrated. CMM is integrated to evaluate the maturity of blockchain implementation. These three models are combined with blockchain-specific attributes derived from the taxonomy proposed by Xu et al., leading to a set of 15 factors.

The weighting of these factors can vary from use case to use case. Therefore, within the frame of this work, no definitive answer can be given as to how each criterion is to be weighed individually. Consequently, it is left to the users to determine the concrete weighting of the criteria. As for electronic bills of lading the criteria *security*, *reliability*, and *scope* would be most important, as it is a document of title [44]. Therefore, these criterions would be weighted higher as other ones.

There are several methods to support multi-criteria decision (MCDA) processes. Well-known approaches for this are analytical hierarchy processing (AHP), PROMETHEE, and analytic network process (ANP). The criteria presented in this paper serve as basis for multi-criteria decision processes to select blockchain applications.

7. References

[1] M. E. Peck, "Blockchain world - Do you need a blockchain? This chart will tell you if the technology can solve your problem," *IEEE Spectr.*, vol. 54, no. 10, pp. 38–60, 2017.

[2] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain Technology Overview," 2018. Accessed on: Mar. 07 2019.

[3] X. Xu *et al.*, "A Taxonomy of Blockchain-Based Systems for Architecture Design," in *ICSA 2017: 2017 IEEE International Conference on Software Architecture: proceedings: 3-7 April 2017, Gothenburg, Sweden*, Gothenburg, Sweden, 2017, pp. 243–252.

[4] Coinmarketcap, *All Cryptocurrencies.* [Online] Available: https://coinmarketcap.com/all/views/all/. Accessed on: Apr. 29 2019.

[5] F. Glaser and L. Bezzenberger, "Beyond Cryptocurrencies - A Taxonomy of Decentralized Consensus Systems," in *Proceedings of the 23rd European Conference on Information Systems, ECIS 2015, Münster, Germany, May 26-29*, 2015, 18 S.

[6] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[7] K. Wust and A. Gervais, "Do you Need a Blockchain?," in *2018 Crypto Valley Conference on Blockchain Technology: CVCBT 2018 : 20-22 June 2018, Zug, Switzerland : proceedings*, Zug, 2018, pp. 45–54.

[8] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, and V. Santamaria, "To Blockchain or Not to Blockchain: That Is the Question," *IT Prof.*, vol. 20, no. 2, pp. 62–74, 2018.

[9] F. Wessling, C. Ehmke, M. Hesenius, and V. Gruhn, "How much blockchain do you need?," in *2018 ACM/IEEE 1st International Workshop on Emerging Trends in Software Engineering for Blockchain: WETSEB 2018 : 27 May 2018, Gothenburg, Sweden : proceedings*, Gothenburg, Sweden, 2018, pp. 44–47.

[10] G. Fridgen *et al.*, "Developing an Evaluation Framework for Blockchain in the Public Sector: The Example of the German Asylum Process," 2018. Accessed on: Mar. 15 2019.

[11] B. Notheisen, S. Willrich, M. Diez, and C. Weinhardt, "Requirement-driven Taxonomy Development – A Classification of Blockchain Technologies for Securities Post-Trading," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019, pp. 4615–4624.

[12] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in software quality: Concepts and Definitions of Software Quality," GENERAL ELECTRIC COMPANY 1, 1977.

[13] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative Evaluation of Software Quality," in *Proceedings of the 2Nd International Conference on Software Engineering*, 1976, pp. 592–605.

[14] R. G. Dromey, "A model for software product quality," *IIEEE Trans. Software Eng.*, vol. 21, no. 2, pp. 146–162, 1995.

[15] R. B. Grady and D. L. Caswell, *Software metrics: Establishing a company-wide program*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[16] *INTERNATIONAL STANDARD ISO/IEC 25010*, 25010, 2017.

[17] D. Samadhiya, S.-H. Wang, and D. Chen, "Quality models: Role and value in software engineering," in *2nd International Conference on Software Technology and Engineering (ICSTE), 2010: 3 - 5 Oct. 2010, San Juan, Puerto Rico, USA ; proceedings*, San Juan, PR, USA, 2010.

[18] A. Adewumi, S. Misra, N. Omoregbe, B. Crawford, and R. Soto, "A systematic literature review of open source software quality assessment models," (eng), *SpringerPlus*, vol. 5, no. 1, p. 1936, 2016.

[19] I. Samoladas, G. Gousios, D. Spinellis, and I. Stamelos, "The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation," in *IFIP – The International Federation for Information Processing*, vol. 275, *Open Source Development, Communities and Quality: IFIP 20th World Computer Congress, Working Group 2.3 on Open Source Software, September 7-10, 2008, Milano, Italy*, pp. 237–248.

[20] QSOS, "Qualification and Selection of Open Source software (QSOS)," 2013. [Online] Available: http://dist.qsos.org/qsos-2.0_en.pdf. Accessed on: Mar. 29 2019.

[21] M. Soto and M. Ciolkowski, "The QualOSS open source assessment model measuring the performance of open source communities," in *3rd International Symposium on Empirical Software Engineering and Measurement, 2009* Lake Buena Vista, FL, USA, 2009, pp. 498–501.

[22] M. Ciolkowski and M. Soto, "Towards a Comprehensive Approach for Assessing Open Source Projects," in *Lecture notes in computer science*, vol. 5338, *Software process and product measurement: International conferences, IWSM 2008, MetriKon 2008,*

and Mensura 2008, Munich, Germany, November 18 - 19, 2008 ; proceedings, pp. 316–330.

[23] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, "Capability maturity model, version 1.1," *IEEE Softw.*, vol. 10, no. 4, pp. 18–27, 1993.

[24] C. Gresse von Wangenheim, J. Carlo, J. Hauck, C. Salviano, and A. von Wangenheim, "Systematic Literature Review of Software Process Capability/Maturity Models," *10th International SPICE Conference on Software Process Improvement and Capability Determination, SPICE 2010*, 2010.

[25] J. Mattila, "The Blockchain Phenomenon: The Disruptive Potential of Distributed Consensus Architectures," *ETLA Working Papers*, no. 38, http://pub.etla.fi/ETLA-Working-Papers-38.pdf, 2016.

[26] B. Singhal, G. Dhameja, and P. S. Panda, *Beginning Blockchain*. Berkeley, CA: Apress, 2018.

[27] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of Consensus Protocols on Blockchain," in vol. 4, *International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2017.

[28] C. Cachin and M. Vukolić, "Blockchain Consensus Protocols in the Wild," Jul. 2017. [Online] Available: http://arxiv.org/pdf/1707.01873v2.

[29] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," in *2017 IEEE International Congress on Big Data - BigData Congress 2017: 25-30 June 2017, Honolulu, Hawaii, USA : proceedings*, Honolulu, HI, USA, 2017, pp. 557–564.

[30] G. Wood, "Ethereum Yellow Paper: a formal specification of Ethereum, a programmable blockchain," Accessed on: Mar. 06 2019.

[31] S. Gubler, *Hyperledger Indy Graduates To Active Status; Joins Fabric And Sawtooth As "Production Ready" Hyperledger Projects.* [Online] Available: https://www.hyperledger.org/blog/2019/04/10/hyperledger-indy-graduates-to-active-status-joins-fabric-and-sawtooth-as-production-ready-hyperledger-projects. Accessed on: Apr. 30 2019.

[32] A. Adewumi, S. Misra, and N. Omoregbe, "Evaluating Open Source Software Quality Models Against ISO 25010," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, LIVERPOOL, United Kingdom, Oct. 2015 - Oct. 2015, pp. 872–877.

[33] Y. Hu, J. Zhang, X. Bai, S. Yu, and Z. Yang, "Influence analysis of Github repositories," (eng), *SpringerPlus*, vol. 5, no. 1, p. 1268, 2016.

[34] C. G. von Wangenheim *et al.*, "Creating Software Process Capability/Maturity Models," *IEEE Softw.*, vol. 27, no. 4, pp. 92–94, 2010.

[35] G. Hileman and M. Rauchs, "GLOBAL BLOCKCHAIN BENCHMARKING STUDY," Cambridge Centre for Alternative Finance, 2017. [Online] Available: https://www.ey.com/Publication/vwLUAssets/ey-global-blockchain-benchmarking-study-2017/$File/ey-global-blockchain-benchmarking-study-2017.pdf. Accessed on: Jun. 26 2019.

[36] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online] Available: https://bitcoin.org/bitcoin.pdf. Accessed on: Mar. 21 2019.

[37] Hyperledger Fabric [Online] Available: https://hyperledger-fabric.readthedocs.io/en/release-1.4/blockchain.html#what-is-hyperledger-fabric. Accessed on: May 24 2019.

[38] Bitnodes, Bitnodes. [Online] Available: https://bitnodes.earn.com/dashboard/?days=365. Accessed on: Aug. 27 2019.

[39] Ethernodes, ethernodes.org. [Online] Available: https://www.ethernodes.org/network/1. Accessed on: Aug. 27 2019.

[40] G. Wood, "Ethereum Yellow Paper: a formal specification of Ethereum, a programmable blockchain," Mar. 2019. Accessed on: Mar. 06 2019.

[41] HACKERNOON, The Blockchain Scalability Problem & the Race for Visa-Like Transaction Speed. [Online] Available: https://hackernoon.com/the-blockchain-scalability-problem-the-race-for-visa-like-transaction-speed-5cce48f9d44. Accessed on: Aug. 27 2019.

[41] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling Hyperledger Fabric to 20, 000 Transactions per Second," CoRR, vol. abs/1901.00910, 2019.

[42] A. Hertig, How Will Ethereum Scale? [Online] Available: https://www.coindesk.com/information/will-ethereum-scale. Accessed on: Aug. 27 2019.

[43] BitcoinCore, Running a Full Node: Support the Bitcoin network by running your own full node. [Online] Available: https://bitcoin.org/en/full-node#what-is-a-full-node. Accessed on: Aug. 27 2019.

[44] S. Wunderlich, and D. Saive, „The Electronic Bill of Lading," in *International Congress on Blockchain and Applications*, Cham, Springer, 2019, pp. 93-100.