applied sciences

MDPI

Review

Software Engineering Applications Enabled by Blockchain Technology: A Systematic Mapping Study

Selina Demi, Ricardo Colomo-Palacios * and Mary Sánchez-Gordón

Department of Computer Science, Østfold University College, 1783 Halden, Norway; selina.demi@hiof.no (S.D.); mary.sanchez-gordon@hiof.no (M.S.-G.)

* Correspondence: ricardo.colomo-palacios@hiof.no; Tel.: +47-6921-5000

Abstract: The novel, yet disruptive blockchain technology has witnessed growing attention, due to its intrinsic potential. Besides the conventional domains that benefit from such potential, such as finance, supply chain and healthcare, blockchain use cases in software engineering have emerged recently. In this study, we aim to contribute to the body of knowledge of blockchain-oriented software engineering by providing an adequate overview of the software engineering applications enabled by blockchain technology. To do so, we carried out a systematic mapping study and identified 22 primary studies. Then, we extracted data within the research type, research topic and contribution type facets. Findings suggest an increasing trend of studies since 2018. Additionally, findings reveal the potential of using blockchain technologies as an alternative to centralized systems, such as GitHub, Travis CI, and cloud-based package managers, and also to establish trust between parties in collaborative software development. We also found out that smart contracts can enable the automation of a variety of software engineering activities that usually require human reasoning, such as the acceptance phase, payments to software engineers, and compliance adherence. In spite of the fact that the field is not yet mature, we believe that this systematic mapping study provides a holistic overview that may benefit researchers interested in bringing blockchain to the software industry, and practitioners willing to understand how blockchain can transform the software development industry.

Keywords: software engineering; blockchain technology; smart contracts; systematic mapping

Citation: Demi, S.; Colomo-Palacios, R.; Sánchez-Gordón, M. Software Engineering Applications Enabled by Blockchain Technology: A Systematic Mapping Study. *Appl. Sci.* **2021**, *11*, 2960. https://doi.org/10.3390/app11072960

Academic Editor: Andrea Prati

Received: 2 March 2021
Accepted: 23 March 2021
Published: 25 March 2021

1. Introduction

Currently, organisations worldwide are showing an increasing interest in blockchain technology due to the promise of significant business benefits. Blockchain technology gained popularity after the publication of the Bitcoin white paper in 2008 [1]. The utility of blockchain, as the underlying technology of Bitcoin, consists of enabling the peer-to-peer exchange of cryptocurrencies, without the involvement of a trusted third party. In 2013, Ethereum was introduced as a platform that incorporated a set of new and promising features to apply the advantages of blockchain to other fields [2]. Ethereum allows building decentralized applications, ranging from financial applications, semi-financial applications, such as self-enforcing rewards for solutions to computational tasks, to non-financial applications, such as decentralized governance and online voting [3]. This is possible due to Ethereum's built-in Turing-complete programming language, which enables anyone to write smart contracts, and to create their own ownership rules and formats of transactions. In 2015, the Linux Foundation launched the Hyperledger project to encourage the development of permissioned blockchains that allow a restricted set of known and identified participants to participate in the network. In this way, secure interactions among participants that share a common goal but do not fully trust each other can be achieved, for instance, businesses that exchange goods or information [4].

The intense hype around blockchain technology and its adoption in different industries [5] has brought the attention of researchers to its application in software engineering

(SE). In 2017, Porru et al. [6] identified the need for specific software engineering practices that consider the distinctive properties of blockchain and coined the term blockchain-oriented software engineering (BOSE). These authors revealed the following BOSE challenges: the need for new professional roles, specific methodologies to ensure security and reliability, new modeling languages, and specific metrics adapted to BOSE, e.g., metrics to measure complexity, resource consumption and performance of such systems. In 2018, the need for the new discipline of BOSE was also advocated by Destefanis et al. [7]. According to these authors, the reliance on smart contracts on a non-standard software lifecycle, poses issues, such as the difficulty in updating delivered applications or resolving bugs by releasing a new software version. These issues call for BOSE to contribute to testable and reliable smart contract software development. In fact, Marchesi [8] called attention to the bi-directional relationship between software engineering and blockchain technology. Colomo-Palacios [9] also emphasized the importance of cross-fertilization between software engineering and technologies that are hyped, such as machine learning and blockchain technology.

Despite the fact that several secondary studies have examined the wider use of blockchain technology, they do not specifically examine its use in improving SE activities. In this paper, we aim to provide a holistic and comprehensive overview of emerging SE applications enabled by blockchain technology. To address this goal, we carried out a systematic mapping study that categorizes research studies according to their contribution, research type and topic. The latter refers to SE knowledge areas where blockchain has been introduced. Finally, we discuss the main findings in order to identify opportunities for future research in the SE field. Therefore, this study can be of interest to two main readers: researchers interested in bringing blockchain to the software industry in the form of applications, and practitioners willing to understand how blockchain can transform the software industry.

The remainder of the paper is structured as follows: In Section 2, a review of the SE, blockchain technologies, and related works is presented. Section 3 describes the research design adopted for this study. In Section 4, we present the main results which are further discussed in Section 5, along with the main validity threats. Finally, in Section 6 the work is concluded and directions are provided for future research.

2. Background and Related Works

2.1. Software Engineering

Over the last 60 years, software has evolved from being a technological tool for solving specific problems, into becoming an industry, which is ubiquitous in most of today's business processes. According to IEEE Standard 610.12 [10], software engineering is defined as "*the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software*". The guide to the Software Engineering Body of Knowledge (SWEBOK) provides a detailed overview of the main SE knowledge areas (KAs) [11], which are also considered by this study. KAs are groupings of information with a related theme, such as software requirements, software process, software testing, software quality, software maintenance, software configuration management and engineering management. There are also some SE practices and their related challenges that deserve to be mentioned due to the emergence of new technologies, such as blockchain that can help to address.

Global software engineering is becoming a more common practice as software products are the result of collaborations among a variety of partners during conceptualization, development, production and maintenance phases [12]. However, it has been reported that many software organizations that adopted global software engineering failed to leverage its benefits in terms of time, cost and skillful resources [12,13]. By conducting a systematic literature review, Niazi et al. [13] identified challenges related to client and vendor organizations. The authors analyzed 101 papers and identified the following challenges: lack of communication and coordination between distributed teams, improper knowledge

transfer which leads to poor quality of software artifacts and lack of team awareness, lack of project visibility, and lack of trust between distributed teams.

Agile practices have been reported to enhance projects' visibility and transparency by means of iteration/sprint planning meetings, standup meetings and retrospectives [14]. The transparency and visibility enhance vigilance which in turn increases trust within teams. However, when scaling up agile practices, technical issues related to inter-team coordination arise [15]. The emphasis on autonomous teams may cause technical divergences, for instance in coding styles, and distrust between teams [15]. Moreover, it is worthy to note that short and frequent release cycles are enabled by continuous integration (CI) practices. However, it has been reported that CI systems are prone to misconfigurations and security attacks, for instance, malicious code can be deployed through the deployment pipeline [16].

Collaboration practices in interorganizational software projects may be complicated, as well. It has been reported that there is an underlying conflict between the common project goal and independent organizational goals, for instance, one organizational goal could be to not disclose functional knowledge [17]. This implies restricted access to artifacts at the partner's side, which in turn impedes complete and reliable requirements traceability and hinders the assessment of whether quality gates have been passed [17]. These organizations may outsource their work to a defined or undefined labour to work on a variety of software engineering activities. The latter is named crowdsourced software engineering. This software engineering paradigm has gained growing interest in industry and academia [18]. In a comprehensive survey on crowdsourced software engineering, Mao et al. [18] reported on SE tasks that make use of crowdsourcing and their respective platforms, for instance, Bountify for coding tasks and uTest for software testing tasks. Besides the advantages that crowdsourcing brings to the SE field, such as reduced costs, time and defects, the authors also pointed out unexplored issues, such as communication and coordination issues, intellectual property and data security. This is not surprising, as crowdsourcing makes use of an open call format for participation and task information is accessible by the general public.

2.2. Blockchain Technology

Blockchain technology, known as the "Internet of Individuals", has been considered a revolutionary paradigm [2]. From an architectural perspective, blockchain is a distributed ledger, that stores transactions in an ever-growing chain of blocks [19]. Figure 1 illustrates how each block contains the hash of the previous block, leading to the structure of a linked list [20].

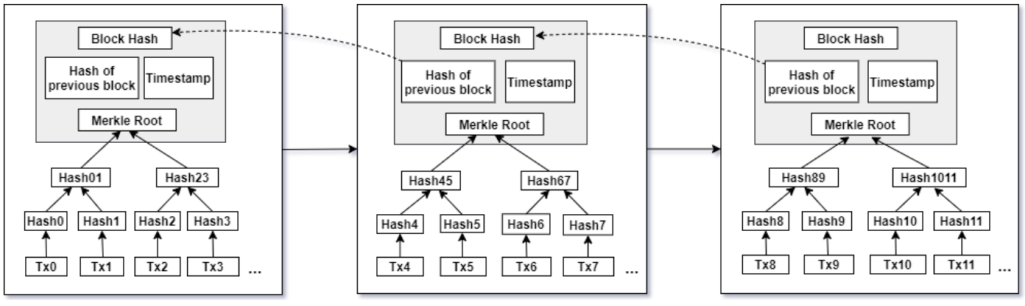

Figure 1. Blockchain structure, adapted from [20].

This structure ensures an important property of blockchain which is *immutability*. The following scenario explains how this property is ensured [21]: Suppose there is an attempt to tamper with data in block n. This would cause a re-computation of that block's hash. Additionally, this hash is also present in the following block $n + 1$. Thereby, the hash of block $n + 1$ would need to be re-calculated, along with all the following blocks' hashes. In

this scenario, creating new valid blocks becomes difficult, as the attacker would need the majority of the network's computational power to rewrite history and calculate new blocks.

Furthermore, blockchain is *decentralized* and as opposed to centralized systems, there is no single entity that manages the network or the blockchain itself [21]. While the decentralized nature of blockchain is advantageous because it eliminates the single point of failure problem experienced in centralized systems, it also implies the need for a distributed validation process. As the number of transactions increases, the computational effort for validation also increases. To address this, blockchain-based systems, such as Bitcoin make use of Merkle trees which are data structures that store hashes of transactions [20]. These hash trees enable nodes to verify transactions without the need to download the entire blockchain [20].

The data stored on the public blockchain are transparent to each node [22]. However, *transparency* comes at the price of privacy. One may argue that the openness of transaction data may imply identifiability. To provide *anonymity*, users are linked to public addresses. As a consequence, users' personal details are preserved to some extent from being revealed, although means to link information in a public blockchain exist. A variety of techniques and proposals to increase the level of anonymity have been extensively discussed in a technical survey provided by Tschorsch and Scheuermann [20].

The potential of blockchain has been greatly extended with the introduction of *smart contracts* [23]. Smart contracts are scripts stored on blockchain, that are triggered by posting a transaction to the blockchain [24]. Afterwards, smart contracts are self-executed in a predefined way on every network node [25]. The result of such execution causes a status change in the blockchain [24]. To simplify the concept, smart contracts allow us to convert the business logic in code. Originally, smart contracts were conceived to automatically achieve agreement between two parties when they sign a contract [26]. To date, the scope of smart contracts has been extended and in fact, from a conceptual perspective, they can perform any task that general-purpose software programs can perform. However, it is important to identify SE use cases that could benefit from the aspects that blockchain offers, e.g., *decentralized*, *immutability*, *transparency*, *anonymity* and *smart contracts*.

2.3. Related Works

Several (secondary) studies have reviewed the application of blockchain, e.g., applications [21] and smart contract development [24]. One of the most recent systematic mapping studies on blockchain technologies was performed by Bharadwaj et al. [21]. In this study, the authors aim to identify and map various domains of research related to blockchain and recognize possible directions for future research. Moreover, Vacca et al. [24] conducted a systematic literature review of blockchain and smart contract development. In particular, the authors identified methods, techniques, tools and challenges faced during the development and testing of blockchain-oriented software. Their analysis suggests future research on how to adapt standard testing techniques to blockchain-oriented software and how to measure code metrics for code optimization. Both previous studies answer questions related to the wider use of blockchain technology, but they do not examine specifically its use in improving SE activities. Indeed, they did not take a close look at the contributions that blockchain aspects can bring to SE.

Specifically, in relation to the application of blockchain to SE, to the best of our knowledge, there appear to be very limited secondary studies. The more closely-related study is a systematic mapping study conducted by Tariq and Colomo-Palacios [27]. This study reported on the uses of blockchain in software engineering and outlined the benefits that this new technology can bring to the SE field. The results of this study indicate that smart contacts can automate the verification of tasks that usually require human-in-the-loop. Smart contracts execute tests, produce results and automatically reward software engineers. Additionally, blockchain can enhance the trust between parties in outsourcing software development. The software requestor uploads the work and reward and the interested developers develop the code and get the reward if all the tests passed, without

the need of any intermediary company. Finally, blockchain can be used to keep track of developers who add third-party components to the final product. In distributed software development, developers may add third-party codes without reporting to team leaders, which may affect software quality and the reputation of the company. By keeping track of third-party components, it is possible to verify their adherence to license compliance policies. In our view, this study provides valuable insights, but it is limited to studies published up to 2018. Indeed, the field of research in relation to blockchain is rapidly evolving, which indicates the need for an updated summary of the most recent research works, in particular, blockchain aspects including but not limited to smart contracts, in order to guide new research activities.

3. Research Design

We carried out a systematic mapping study according to the guidelines for systematic mappings in SE proposed by Petersen et al. [28]. This section describes the design of the proposed research process which includes: (a) defining research questions, (b) conducting the search for relevant papers, (c) screening of papers, (d) keywording of abstracts, and (e) data extraction and mapping. Figure 2 depicts the essential process steps.

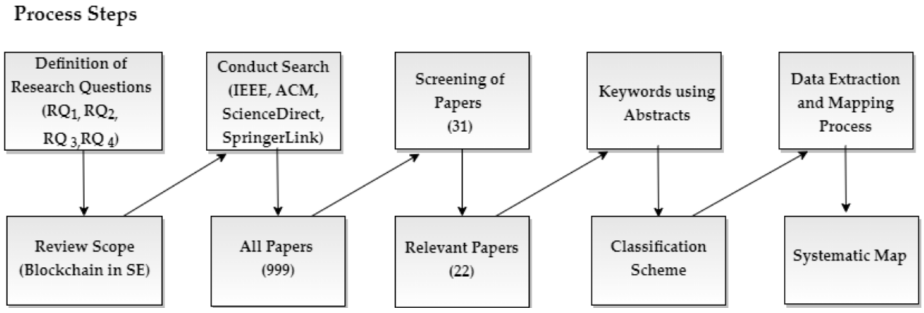

Figure 2. Overview of the systematic mapping study, adapted from [28].

3.1. Research Questions

The goal of the study is to provide a comprehensive overview of how blockchain technology has been used in SE. According to the goal, we formulated four research questions (RQ), as follows:

1. RQ_1 *What is the trend of studies that use blockchain in SE?* With this research question, we aim to provide a quantitative overview of the current research progress on the uses of blockchain technology in SE. This research question will also look into research methods in order to provide insights on the trends of the research approaches adopted by the selected studies.
2. RQ_2 *What are the blockchain uses in SE that have been reported in literature?* To answer this research question, we classify the studies according to SE knowledge areas in which blockchain can contribute. This, in turn, will help in identifying potential areas that have been overlooked.
3. RQ_3 *What blockchain platforms are used in developing SE applications?* This research question aims to provide implementation details of blockchain-enabled SE applications, regarding blockchain platforms and consensus algorithms. These findings may suggest which blockchain platform is the most suitable for specific SE use cases.
4. RQ_4 *How can blockchain contribute to the SE landscape?* With this research question, we intend to provide a holistic view of the contributions that blockchain can bring to the SE landscape. To achieve this goal, we map blockchain properties with SE challenges addressed.

3.2. Search Process

The search process aims to identify as many primary studies related to the research questions as possible. To do so, we develop a research protocol that included an unbiased

search strategy. Our strategy to identify potential primary studies was based on the scope of our study, i.e., *blockchain* technology in *software engineering*. Thereby, these keywords were used as search terms in order to construct the search string. For the automated search, the search string consisted of two main parts: *blockchain* AND *software engineering*. Furthermore, a list of alternate terms was used and connected through the Boolean operator OR to construct a broader search string. Blockchain technology is per se a *distributed ledger,* and we argue that its main uses are enabled by smart contracts. We conducted trial searches with these keywords and observed that relevant studies, such as [29] were only retrieved when smart contracts were used in the search string. In fact, it has been reported that smart contracts have the strongest implications in software engineering [30]. In a wider scope, smart contracts have been considered as the technology with the highest potential to revolutionize transactions among people and businesses [31]. Therefore, to ensure a broad set of results, we included smart contracts in our search string due to their potential, particularly in software engineering.

Moreover, by iterating with different versions of the search string, we observed that some authors mention only software development in their studies although software engineering encompasses the concept of *software development*. Therefore, this term was also included, to minimize the risk of missing relevant literature. As a result, we constructed the following final search string:

(blockchain OR "smart contracts" OR "distributed ledger") ***AND***

("software development" OR "software engineering")

We performed an automated search process using four digital databases: (a) IEEE Digital Library, (b) ACM Digital Library, (c) Science Direct, and (d) Springer Link. The selected databases are well-known and are constantly used in secondary studies in the software engineering field [28,32]. The main filter that we used is the field of "Computer Science" and subfield of "Software Engineering", where available in the databases. Regarding the time period, we searched for studies published up to 2020. That means that the search period did not have a lower bound. The details of the primary studies (i.e., title, author(s), abstract, keywords, year of publication, and the name of the data source) were directly exported from the digital libraries to a reference manager, namely Zotero.

3.3. Screening of Studies for Inclusion/Exclusion Criteria

The initial search process returned a set of 999 studies. We analyzed the title, abstract and introduction/conclusion (when necessary) of these studies against the inclusion/exclusion criteria. On a second round, the inclusion/exclusion criteria were applied to each study's full text. Table 1 lists the inclusion/exclusion criteria used in the manual inspection.

Table 1. Inclusion and exclusion criteria.

Inclusion Criteria	Exclusion Criteria
Studies that focus on the uses of blockchain in SE	Studies that use blockchain in domains other than software engineering
If a journal study follows the same conference study, only the journal study is included	Studies that focus on SE issues of blockchain-based software
If a similar study is published by more than one source by the same authors, only the most recent or extended version is included	Studies that do not discuss or propose approaches for using blockchain in SE
The full text of the study is available in English	The study is an editorial, keynote, tutorial, poster or panel

There were a significant number of the initial studies focused on SE for blockchain-based software, which is outside the scope of this research, e.g., [6,7,33]. Other studies adopted principles behind blockchain to specific domains such as cyber–physical systems (CPS) and embedded systems. Although software is an important component of these systems, it has been reported that software engineering and development in such systems are tailored to the complex nature of CPS [34]. Therefore, studies such as [35] were excluded,

because we decided to take a more general domain-agnostic approach. Furthermore, few studies were excluded because their respective extended studies were identified. For instance, [36] was the initial version of [37]; in such a case the extended version [37] was chosen (for excluded studies refer to data available online [38]). As a result of the screening process, only 18 studies fit our inclusion criteria. Additionally, backward snowballing was carried out, given that we delimited our initial search to four databases. References of the selected studies were scanned and four relevant studies were identified. A total of 22 research papers constitute the final set of our primary studies (see Appendix A). Table 2 shows the number of studies selected in each of the phases of the research process.

Table 2. Overview of primary studies.

Database	Initial Search	First Exclusion	Final Exclusion
IEEE Xplore	140	17	10
ACM Digital Library	339	13	7
Science Direct	320	0	0
SpringerLink	200	1	1
Selected studies	999	31	18
Snowballing			4
Primary studies			22

3.4. Classification Scheme

We opted for the keywording technique, in order to develop the classification scheme, as suggested by Petersen et al. [28]. The first step to develop such a scheme consists of carefully reading the abstracts of the selected studies and extract keywords related to the research topic, research type and contribution type. These keywords were then clustered to form map categories. The three facets are as follows:

- Research topic facet. This facet intends to structure the topics related to the uses of blockchain technology in the SE KAs. The main topics are then discussed in combination with the research type and contribution facets.
- Research type facet. We aim to identify the research approach adopted by the selected studies. The classification provided by Petersen et al. [28] was used, due to its generalizability and simplicity. According to this classification, the main research types are validation research, evaluation research, solution proposal, philosophical, opinion and experience study. These research approaches are presented and explained in Table 3.
- Contribution type facet. We aim to investigate the contribution type of the selected studies. We adapted the contribution type facet introduced by Petersen et al. [28], as explained in Table 4.

Table 3. Research types.

Research Type	Explanation
Validation Research	Novel approaches are validated in experimental settings, but they are not implemented in practice.
Evaluation Research	Novel approaches are implemented in industrial settings and this implementation is evaluated. It also includes studies that evaluate the impact of implementing blockchain technology in SE.
Solution Proposal	The study identifies a problem and proposes the respective solution. Its applicability, benefits and challenges are discussed, yet neither validated nor evaluated.
Philosophical Study	The study presents a new perspective at existing issues, usually by means of a conceptual framework.
Opinion Study	The study presents the opinion of the author related to a specific topic.
Experience Study	The study represents the personal experience of the author related to the practical implications of a specific technique or approach.

Table 4. Contribution Types.

Contribution Type *	Explanation
Model	Studies that propose a novel model or improve existing ones.
Framework	Studies that propose a conceptual framework to guide the development of a solution.
Interviews	Studies that aim to explore a particular technology such as blockchain in the field of SE, by means of interviews.
Platform	Studies that propose platforms that add value to existing business models or establish new ones.

* The contribution type of the studies is based on what the authors of these studies claim to contribute (in cases where they specify their contribution type).

4. Results

4.1. Trend of Studies That Use Blockchain in Software Engineering (RQ_1)

The interrelation between blockchain and SE, in particular, the applications of blockchain in SE have been overlooked according to Beller and Hejderup [30]. This is also supported by the low number of studies identified in our systematic mapping. We retrieved a total of 999 studies by searching four online databases. These studies went through a two-stage assessment process against inclusion/exclusion criteria and as a result, 18 studies were selected. This indicates a high number of irrelevant studies retrieved (98% noise), which is common in database searches [39]. We observed some of the irrelevant studies to understand the main reasons for exclusion. Studies were excluded in the first assessment phase mainly because they explore blockchain in other more mature fields, such as supply chain and healthcare. Additionally, studies were excluded in the second phase mainly because they focus on software engineering issues in blockchain-oriented software, which is outside the scope of our study. We also carried out snowballing, as complementary to the database search. In total, we selected 22 studies. A plausible explanation of the low number of studies could be related to the novelty of blockchain technology itself. While it is true that Bitcoin, the first blockchain application, was invented in 2008 by Satoshi Nakamoto, smart contracts became popular with the release of Ethereum in 2015, as a Turing-complete blockchain platform. A recent systematic mapping study on blockchain-based smart contracts carried out by Macrinici et al. [40] found out an increasing trend of publications since 2016.

The first study that we identified was published in 2015 and it uses cryptocurrency blockchain technology for decentralized software license validation [41]. As expected, this study does not make use of smart contracts. The first study that implements smart contracts was published in 2017 with the goal to develop a non-stoppable virtual organization for software development communities [29]. The remaining studies (91%, 20/22) were published during the last three years (2018–2020) and they were devoted to addressing issues in collaborative software development, by means of blockchain technology. This trend indicates growing research efforts in blockchain-based software engineering (see Figure 3). The growing trend seems to respond to Marchesi's [8] call for more studies in the 1st International Workshop on Blockchain Oriented Software Engineering (BOSE). Moreover, the selected studies were published in conferences and workshops (see details in Appendix A), which suggest a conference-driven publication culture.

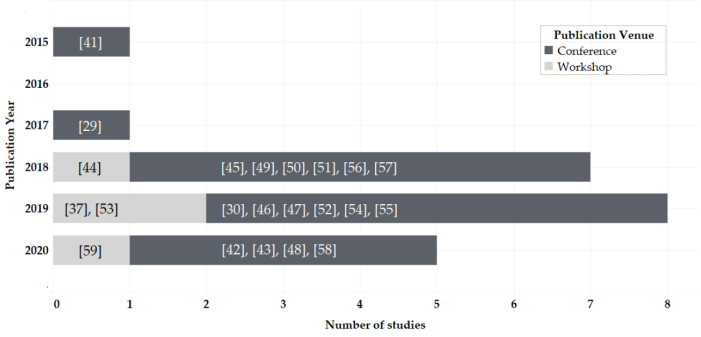

Figure 3. Distribution of studies based on publication year and publication venue.

Figure 4 shows that all the studies except [42] propose solutions or validate them in experimental settings, while 17 out of 22 studies contribute by means of models or frameworks. Many of the studies (10 out of 22) propose solutions that need to be implemented and tested to assess their strengths and weaknesses. We identified only one evaluation study [42]. The authors of this study investigate the impact of blockchain in global software development, by conducting interviews with industry practitioners. However, the number of interviews is limited, only 5. The scarce empirical evidence on the impact of blockchain in SE calls for empirical research on the influence and repercussion of blockchain in SE.

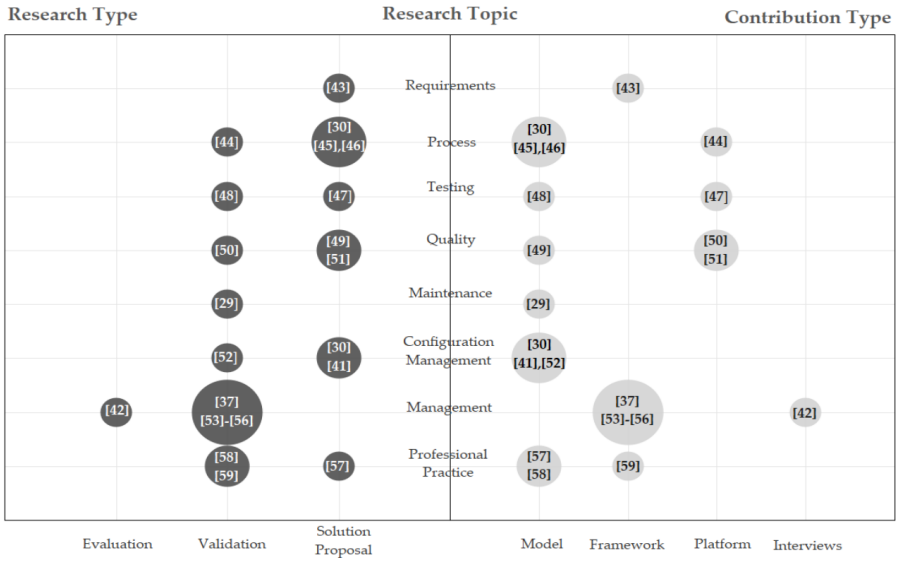

Figure 4. Visualization of our systematic map in the form of a bubble plot.

4.2. Blockchain Uses in Software Engineering Reported in Literature (RQ_2)

This section presents the findings of our study with respect to the uses of blockchain to support software engineering activities. Figure 4 is a scatter plot that depicts the intersection of the research type and contribution type facets with the research topic facet. Bubbles' sizes represent the frequency of studies that fall within these intersections (The total number of studies in this map overcomes the number of selected studies, because one study, in specific, [30] refers to more than one topic). Inspired by SWEBOK, we classified the selected studies into 8 categories, namely, (*i*) software requirements [43], (*ii*) software engineering process [30,44–46], (*iii*) software testing [47,48], (*iv*) software quality [49–51], (*v*) software maintenance [29], (*vi*) software configuration management [30,41,52], (*vii*) software engineering management [37,42,53–56], and *(viii)* professional practice [57–59]. We briefly describe the SE applications below.

Software requirements. Requirements management and traceability face a variety of challenges, such as lack of confidence in existing tools, integration issues among heterogeneous tools, manual work, lack of motivation, and confidentiality constraints that impede complete traceability across organizational boundaries [43]. In this regard, a blockchain-based approach was proposed for the trustworthy management and traceability of requirements in interorganizational software projects [43]. Stakeholders register, requirements and metadata on the blockchain and track their evolution through blockchain query functions. Blockchain enables an auditable history of requirements that is visible and verifiable by authorized users, such as partners and customers. Although this study is part of ongoing work, the author claims that blockchain has the potential to enhance the immutability, trust, visibility and traceability of requirements throughout the software development lifecycle (SDLC).

Software engineering process. Król et al. [44] proposed a reliable platform called ChainSoft, for outsourcing software development. This platform makes use of oracles that allow the smart contract to communicate with GitHub/Travis CI. The software requestor

creates tests, submits them to a GitHub repository, and submits the task and reward to a smart contract. The developer creates code that passes the tests and uploads the solution to his/her GitHub repository. The smart contract then checks whether all the tests are covered by the solution. It is noteworthy that this is the only study that performs a security analysis. According to this analysis, the platform relies on the honesty of third-parties, which raises several concerns: *(i)* it may happen that GitHub/Travis CI do not run the tests as intended, intentionally *(ii)* their services may become unavailable which leads to users submitting tasks at a later time and paying additional transaction fees, and *(iii)* relying on these systems introduces centralization to some extent, which goes against the decentralized blockchain mindset.

Lenarduzzi et al. [45] introduced two blockchain-based models for the management of Scrum-based and Lean–Kanban projects. According to these authors, smart contracts enable the automation of the acceptance phase and the payment to developers. The customer creates the smart contract for the specific project and the product owner registers user stories, acceptance tests and the hash of expected output. Developers execute the acceptance tests and if the tests pass, they register the hash of the correct output. The smart contract checks the hash of the output against the hash of the expected output. If all the tests passed, Ethers are sent automatically to the developer's address. Although this proposal was not validated, the authors remain optimistic about the potential of using blockchain and in specific smart contracts, to transform other phases of the SDLC that currently rely on human rationale.

Yilmaz et al. [46] proposed a blockchain-based model to address integrity in large-scale agile software development. Interestingly, these authors perceive software development as the Byzantine Generals' Problem and software practitioners who cause defects as Byzantine participants. In their proposed model, developers are miners who develop code and testers are the validators of developers' work. The project leader publishes work structs on the blockchain (work description, test description, reward, and the signature of the project leader). The work structs form the genesis block and the consequent blocks consist of additional work structs, code structs (related work ID, code, and signature), and the hash of the previous block. These blocks are validated and digitally signed by testers.

Finally, Beller and Hejderup [30] proposed a decentralized continuous integration model, as opposed to the centralized Travis CI. In this model, developers enter a build and its reward to the network and interested workers perform the build. Once consensus is reached among peers, the transaction is appended to the distributed ledger. The authors raise two concerns regarding the implementation of the system: the storage of build logs (on the blockchain or off-the-chain) and the issue of non-deterministic builds.

Software testing. Wang et al. [47] adopted blockchain technology to address issues in software testing using bug bounties, such as the lack of transparency regarding the price information for bug bounties, and the difficulty in establishing mutual trust between testers and software buyers/sellers. Their proposed solution consists of appending test results and documents as blocks on the distributed ledger to ensure traceability in the context of disputes between the party who initiated the bounty and the testers. Different testers adopt different methods and this may lead to discrepancies between testing results. To resolve this issue, the authors propose a novel consensus protocol, namely Proof of Skill. Once testers upload their results to the platform, their skill rating which relies on the historical testing performance is calculated and then grouped. The dispute is won by the group with the highest skill rating.

Yau and Patel [48] used blockchain to share software testing information, e.g., test planning, test cases, test results and test results assessment, among diverse teams in a reliable and trusted manner. The authors implemented their model using Hyperledger Fabric. According to the lifecycle of a transaction in Hyperledger Fabric (execute-order-validate), each testing result needs to satisfy acceptance criteria, endorsement policy and consensus. This eliminates the risk of injection attacks in software systems and makes the proposed model suitable for trusted testing in large-scale and complex projects.

Software quality. Badreddin [50] developed the Susereum platform using the blockchain development platforms: OpenChain and Quorum. Susereum platform intends to empower software teams to dynamically propose sustainability and quality metrics which are then used to validate sustainable code contributions by miners. According to the authors, sustainability rating can be perceived as a way to assess code quality. The authors also raise questions regarding ways to ensure metrics' quality and ways to measure the impact of this proposal on software sustainability.

Kim and Kim [49] proposed a model that stores code quality measurements on the distributed ledger to ensure their reliability and immutability. The authors measure software quality in terms of code complexity, by calculating the cyclomatic number as a function of the number of edges, number of nodes and the number of connected components, as well as connectivity by measuring interconnections among modules. According to the authors, this model ensures the transparent code quality measurement history, i.e., visible to any authorized user.

Lin et al. [51] proposed a blockchain-based platform namely CoderChain. This platform consists of three roles: developers, code and jury. The novelty of this approach lies in the "jury" concept which entails developers with advanced skills. The jury assesses and reviews the code of developers and stores this review on the blockchain ledger. The authors conclude that such an approach ensures the reliability of code reviews, the anonymity of reviewers and enhances code quality.

Software maintenance. Given that the maintenance of open-source software projects is at the mercy of volunteers [29], it may happen that after development the software is no longer maintained and further developed. To address this issue, Alimoğlu and Özturan [29] proposed implementing a continuously operating virtual organization to represent the software, by means of Ethereum smart contracts. Their model enables the collection of funds for software development proposals through cryptocurrencies and, additionally, records citations, software usage and software executions on the public blockchain. The proposed model is deployed on a local Ethereum blockchain network, however, the validation is constrained to the gas consumption of smart contracts functions and does not consider aspects of performance efficiency, such as latency and throughput.

Software configuration management. Herbert and Litchfield [41] introduced the first peer-to-peer software license validation approach based on blockchain technology back in 2015. Different from the other studies that make use of smart contracts or chaincode, such an approach uses bitcoins that are held by the user to prove software entitlement. Blockchain enables developers or vendors to allocate licenses to users in a cost-effective and transparent manner. These licenses are stored on the blockchain ledger and can be verified by any node.

Beller and Hejderup [30] proposed a user-run package management model, where any participant may propose new packages and verify the work of others. This approach aims to replace centralized package management systems, in order to democratize and professionalize the SE field. In turn, D'mello and González-Vélez [52] proposed a blockchain-enabled package control system that stores metadata of the new packages (owner name, package name, version and dependencies) on the distributed ledger and package files on IPFS (InterPlanetary File System). Their model was successfully tested with 4338 packages from NPM (Node Package Manager). The authors strongly believe that the use of smart contracts could have a significant impact on the open-source ecosystem, in terms of maintaining a block chain of version and dependencies in software packages.

Software engineering management. The development of software by distributed teams that make use of a variety of artifacts from disparate sources leads to issues, such as the lack of integrity, lack of provenance, and ineffective compliance monitoring [37]. To address these issues, Bose et al. [37] proposed a blockchain-based governance framework for trustworthy software development which consists of three layers, as follows:

- *Data Layer*. The goal of this layer is to capture and monitor event data from the variety of tools used throughout the SDLC. To model the diverse sources of data, the authors

propose the use of PROV family of specifications which support the capture of the software development resource view, process-flow view and data-flow view.

- *Analytical Layer*. This layer comprises the analysis of event data for: compliance checking, provenance services and integrity assessment. Regulations and best practices, e.g., libraries should not be used if their vulnerability score is 5 and above (vulnerability threshold is 4), are encoded in the form of smart contracts [53]. Additionally, the framework enables the analysis of provenance through services, such as provenance query services that can focus on agents, artifacts or the process, and inference services to uncover non-trivial insights [54]. Finally, the framework provides services that generate composite cryptographic hashes of artifacts as the result of the concatenation of the hashes based on metadata and hashes based on content. These sub-identities uniquely identify artifacts, and their combination generates the composite software identity [55].
- *Advisory Layer.* This layer provides services to alert users in case of non-compliant issues and suggests remedial measures to address such issues.

Ulybyshev et al. [56] addressed the problem of unauthorized access, modification and transfer of software modules among entities in collaborative software development, by means of blockchain technology. Their proposal relies on access-control policies established in smart contracts. Software module requests and transfers are recorded on the blockchain ledger, ensuring in this way the integrity of the provenance data and accountability.

More recently, an evaluation study was carried out by Akbar et al. [42] to explore the impact of blockchain on the global software development environment from a management perspective. The authors of this study conducted interviews with academic experts and industry practitioners. Their findings confirmed the positive impact of blockchain in such environments in the following dimensions: visibility, updated task status, secure payment at distributed sites, and accountability of team members.

Software engineering professional practice. Jhala et al. [57] proposed a smart collaboration mechanism where each *collaboration* is encoded as a transaction message. Changes that collaborators make are appended to the blockchain after consensus from all the parties. Their proposed model consists of three entities: collaborators who perform code commits, administrators who authorize specific actions, and miners who verify transactions. One of the major drawbacks of this system is increased latency, as messages are broadcasted first to administrators and then to miners.

Yau and Patel [58] address the issue of *trusted coordination* in collaborative software development. Each software team is represented by a blockchain node and generates software specifications for each of the SDLC phases, e.g., requirements, implementation and testing, in the form of smart contracts. The smart contract contains teams allowed to participate in the smart contract, and software specifications in {key, value} format. Software specifications are then assessed against the results generated by the other allowed teams. The authors provide a couple of examples to illustrate their approach and plan to analyze the scalability in complex and large-scale software projects.

Singi et al. [59] proposed a blockchain-based incentive framework to ensure transparent *incentives to software engineers* who contribute to any activities throughout the entire software lifecycle: pre-development, development, post-development. The framework captures events and assets' metadata from the distributed software delivery ecosystem and analyzes these events according to incentive policies. In case of adherence, valid participants are incentivized by means of digital tokens. As future work, the authors proposed the incorporation of gamification elements to enhance the incentivization of software engineers.

4.3. What Blockchain Platforms Are Used in Developing SE Applications (RQ_3)

As can be seen in Table 5, Ethereum and Hyperledger Fabric are the most used blockchain platforms by the selected studies. This is not surprising, given that these platforms are designed to run smart contracts or chaincode, and the ability to run code has the

strongest contributions for blockchain-oriented SE according to Beller and Hejderup [30]. In Ethereum, all programmable computations, such as invoking and deploying smart contracts, are subject to fees [60], which serve to reward miners for adding blocks to the system. Two of the selected studies measured the fees for calling smart contracts functions [29,44]. Krol et al. [44] evaluated a cost of 1 $ up to 8 $, depending on the processing speed. They concluded that a cost below $4 is reasonable in the case of software development projects that usually require a significant amount of money to finish. On the other side, Hyperledger Fabric provides higher levels of scalability and confidentiality due to the delineation of roles, such as ordering and validating transactions, which in turn allows for configurable consensus. However, challenges related to Hyperledger Fabric may be the high communication overhead and potential DoS attacks. Ulybyshev [56] provided a modified transaction validation procedure that involves less communication overhead and protects Hyperledger from DoS attacks. Their proposed workflow eliminates client communication with the ordering module, instead of that endorsers send endorsed transactions directly to the ordering module. However, these authors do not provide information about how consensus is reached regarding the order of transactions.

Table 5. Blockchain platforms used in developing SE applications.

Platforms	Bitcoin	[41]
	Ethereum	[29], [37], [44] *, [52], [53] $^{\Delta}$, [54] $^{¥}$, [55] †, [59]
	Hyperledger Fabric	[48], [49], [56] $^{\Diamond}$, [58]
	Others	Susereum [50]

* ChainSoft, $^{\Delta}$ CAG, $^{¥}$ Blinker, † ShIFt, $^{\Diamond}$ Blockhub.

There is very little evidence regarding consensus mechanisms used in our selected studies. We identified only two consensus mechanisms that were proposed but were not deployed [46,47]. Yilmaz et al. [46] presented a blockchain-based approach in which developers are miners who create code and testers are validators. They also proposed a consensus mechanism to handle situations when many developers claim to have accomplished the same work. The consensus mechanism relies on a probability distribution function which is constructed based on developers' activity (their contribution and waiting time) and testers' preference (number of votes for each code). The probability of the developers' code to be confirmed increases with the increase of developers' contribution, waiting time and votes. Wang et al. [47] presented an approach in which contracted testers test software for potential vulnerabilities, and register testing results on the blockchain. To resolve potential disputes that can occur in the testing process, they proposed a consensus mechanism named Proof of Skill. The idea is to maintain a skill rating for each tester based on the testing job category and previous testing job performance on the platform. The algorithm groups testers based on their votes and computes the average skill rating for each group. The dispute is won by the group with the highest skill rating. Finally, the algorithm increases the skill ratings for testers in the winner group and decreases for those in the loser group.

4.4. Contributions of Blockchain to the Software Engineering Landscape (RQ_4)

In this section, we provide a holistic view of the contributions that blockchain can bring to the SE landscape. Table 6 shows the mapping of blockchain properties and the SE challenges that they address. In what follows, blockchain properties and their contributions to SE dimensions are described.

Table 6. Mapping of blockchain properties and SE challenges addressed.

Blockchain Properties	SE Challenges Addressed	Selected Studies
Decentralization	Single point of failure and compromise problems of centralized systems, e.g., GitHub, Travis CI, and cloud-based package managers	[29,30,41–43,46,48,50,52,57,58]
Transparency and trust	Lack of trust and visibility of SDLC activities among distributed teams that usually operate in silos	[29,30,37,42,43,45–49,51–54,56–59]
Immutability and data security	Unauthorized software artifacts access, modification and transfer	[29,37,41–43,46,48,49,52–56,58,59]
Anonymity	Lack of fairness in reviewing code/software contributions	[51]
Non-repudiation	In outsourcing and crowdsourcing, software requestors or bug bounties intiators may repudiate payments to developers or testers. In collaborative software development, collaborators may repudiate updates they make.	[44,47,48,53,57,58]
Smart contracts/Chaincode	Manual work in the verification of SDLC tasks, e.g., acceptance phase, compliance to best practicies and regulations, and automatic payment to software engineers.	[29,30,37,43–45,47–50,52–56,58,59]

Decentralization. The software development ecosystem relies on centralized systems, such as GitHub and Travis CI [30], and cloud-based package managers [52]. These systems pose the risks of single points of failure and compromise. For instance, in 2015, GitHub has been the target of DDoS (distributed denial-of-service) attacks where a high number of illegitimate requests resulted in intermittent outages [61]. To avoid such issues, these centralized systems can be replaced with decentralized systems which enable all the authorized parties to have the same view on transactions stored on the distributed ledger. Distributed systems offer higher availability and resilience to intermittent outages [30].

Transparency and trust. Distributed teams operate within their own boundaries, which hinders a 360-degree view over the entire software development lifecycle [59]. For instance, distributed teams record compliance-related data in their local repositories [53]. These data can be manipulated prior to sharing with other teams or clients, which introduces trust issues. In turn, the detection of manipulated data impacts delivery schedules, whereas the non-detection may lead to penalties or loss of reputation. Blockchain can serve as the backbone of the software development lifecycle, meaning that all SDLC events and artifacts' metadata can be recorded on the blockchain. In this way, blockchain provides distributed collaborators with a holistic view of the software development lifecycle. The authorized collaborators can verify the trustworthiness of software-related information at any time. Therefore, blockchain can be used as a shared distributed ledger, which due to its transparency and verifiability contributes to mitigating trust issues between different parties involved in the development of a complex and large-scale software project.

Immutability and data security. In collaborative software development, multiple participants can access, modify and transfer software artifacts. Unauthorized software modifications have been considered as the key issue as software crosses teams' boundaries [55]. For instance, participants can modify code with fraudulent intent or add vulnerable open-source components which enable hackers to carry out data breaches. The blockchain structure as a linked list allows to track the history of each software artifact and detect any unauthorized attempts to access, modify or transfer software artifacts. The immutability property of blockchain enhances the security of software artifacts stored on it, such as code, build files, and third-party components, as they are encrypted, hashed and appended in chronological order on the blockchain.

Anonymity. Code hosting platforms such as GitHub allow users to store code and rate the code as a way to reflect code quality. However, this is not enough to assess the quality of code contributions [51]. To address this issue, Lin et al. [51] proposed an approach in which developers with advanced skills, referred to as the jury, review the code of developers in a double-blind process. In this case, anonymity is particularly important to ensure fairness in the reviewing process, which in turn improves the code review process and software quality.

Non-repudiation. Stakeholders in the software development lifecycle can shirk responsibility. For instance, when software is being developed in a collaborative manner, it may occur that a collaborator claims to not have added a specific patch [57]. Additionally, in software-testing with bug bounties, it may occur that bug bounties initiators claim that they discovered the bug and refuse to reward the tester [47]. These issues can be addressed with digital signatures and blockchain consensus mechanisms. Transaction messages containing updates or testing results are digitally signed and validated depending on the consensus algorithm. Once a transaction is recorded on the blockchain, it cannot be repudiated.

Smart contracts/Chaincode. Smart contracts can automate the verification of tasks, which usually involves human-in-the-loop. For instance, smart contracts can be used to automatically verify that tests passed [44,45], and to verify the compliance of SDLC activities to best practices and policies, e.g., security vulnerabilities compliance and third-party license compliance. Additionally, smart contracts can automatically reward software engineers once they have completed their tasks and passed the criteria encoded in smart contracts via digital coins [45] or tokens [59]. The increased automation has positive impacts on the speed, quality, and efficiency of the software development process.

5. Discussion

5.1. Theoretical Contributions and Research Implications

The software development process is considered intrinsically complex, and overlooking such complexity may impact the success rate of software projects [62]. This complexity can be attributed to the fact that nowadays software development is performed globally in collaboration with a variety of participants, such as vendors or crowd-workers. While this collaborative effort intends to improve software quality, reduce costs and time to market [12], previous research has also reported on challenges. These are lack of communication and coordination among distributed stakeholders, lack of trust, lack of project visibility, poor knowledge transfer [13], as well as intellectual property and data security [18]. The alignment between these issues and blockchain properties inspired us to explore the uses of this novel technology in SE. Our findings indicate that blockchain, due to its inherent properties of decentralization, trust, transparency and immutability, can contribute to the software engineering landscape in four main dimensions (See Table 6): *(i)* eliminate single point of failure and compromise, by replacing centralized systems, such as GitHub, Travis CI, and cloud-based package managers, with decentralized systems that offer availability and resilience. *(ii)* blockchain can serve as a backbone of the SDLC ecosystem, in which all software events and artifacts metadata can be stored and accessed by distributed stakeholders. In this way, blockchain enables a holistic view of the entire software lifecycle, which improves trust and collaboration among distributed stakeholders, and facilitates auditability, compliance, provenance and identity assessment analysis. *(iii)* secure software artifacts sharing among collaborators, by detecting any unauthorized attempt to access, modify or transfer software artifacts. *(iv)* smart contracts have the potential to enhance software development efficiency, by automating a set of activities that usually rely on human rationale, such as verifying if tests passed acceptance criteria, verifying if source code passed quality criteria, compliance to best practices and regulations, and enabling automatic payments to software engineers. We did not find evidence of using smart contracts for the automatic assessment of software design and requirements against pre-defined acceptance criteria, which presents a promising research direction.

Our findings reveal that researchers have focused on improving disparate SE knowledge areas, but have not investigated the holistic impacts of blockchain on software development efficiency and quality. Such investigation can foster the implementation of blockchain use cases in software organizations and their evaluation by software practitioners. Based on our findings, there is no study that implements blockchain-oriented SE applications in organizational settings to date. This may be because of the limited research in this area, the nascent phase of blockchain development, and the existence of unresolved technical challenges [63]. More future efforts devoted to this topic in the form

of prototypes and proofs-of-concept can encourage software practitioners to incorporate disruptive blockchain properties into the SE landscape in order to harvest tangible benefits.

Software practitioners should conduct a careful analysis to decide whether blockchain is required and feasible in the context of SE. This analysis should consider the alignment between blockchain principles and specific software development principles and strategies, alternative tools and technologies, appropriate blockchain platforms, and implementation challenges. There is very little evidence regarding these aspects in the extant literature. In what follows, we discuss some of these aspects to guide future research efforts. Agile practitioners should consider the alignment of blockchain principles with agile principles. At first glance, one may argue that the subtle notion of trust in agile practices contradicts the trustless nature of blockchain technology. Agile software development relies on trust among developers, between developers and the product owner, and between the product owner and business stakeholders. This implies no control of the work done but transparency, communication, accountability and collective responsibility [14]. On the other hand, researchers point out that blockchain is trustless [64], which means that there is no need to trust participants, as the system is secure even in the presence of Byzantine participants. In other words, blockchain could enable collective trust on the basis of mutual distrust, the so-called trustless trust [65]. Therefore, blockchain transforms the trust notion from trusting peers to trusting the system. Researchers can further contribute to this topic by mapping, comparing and discussing other blockchain principles with agile principles.

Regarding the selection of blockchain platforms, the main items that should be considered are the network permission and smart contracts support [66]. Our findings indicate that most of the studies use smart contracts to enhance the automation of SDLC activities. Therefore, the most used blockchain platforms by the selected studies are Ethereum and Hyperledger Fabric due to their ability to execute code. Ethereum is a permissionless blockchain platform that can be accessed and verified by anyone, whereas Hyperledger Fabric is a permissioned blockchain platform that allows only a set of predefined participants to join the network [60]. Due to the differences in network permission, we perceive Ethereum as more suitable for the open-source ecosystem with diverse contributors. For instance, in [52] authors use blockchain for managing open-source packages. Making information such as packages and their dependencies, available on a public ledger increases the chances for code reuse. Additionally, Ethereum can be adopted in crowdsourcing projects in which the requestor delegates specific development or testing tasks to the general public, and any developer or tester can compete to solve the task and get the reward. If the collaborators are known, for instance, different departments or teams in a distributed organization, and they need to share software-related information, e.g., [48], a blockchain platform with permissioned accessibility, such as Hyperledger Fabric is more appropriate. As blockchain technology is continuously evolving, we expect blockchain 4.0 platforms that incorporate artificial intelligence to expand the impact of blockchain on SE practices [9]. Researchers can compare the features of different blockchain platforms and discuss their applicability to SE use cases.

Furthermore, practitioners should consider implementation costs, governance, regulative compliance, and efficiency limitations. A few limitations of blockchain-oriented SE applications have been mentioned in one of our studies [48]. These are large setup overhead (blockchain infrastructure setup and cryptographic tools need to be installed in each node) and storage overhead (the same information is replicated in each node). The impact of these limitations on software development efficiency should also be investigated. Future research can be devoted to the investigation of these open issues to guide implementation efforts of blockchain-oriented SE applications. Finally, the emerging field of blockchain-oriented SE introduces the need for professionals with well-defined skills in both blockchain and software engineering [6], in order to ensure that blockchain applications satisfy the requirements of software engineering.

5.2. Threats to Validity

An update of systematic mapping guidelines in SE published in 2015, by Petersen et al. [67] suggested systematic mapping studies to consider the following validity types: theoretical validity, descriptive validity and interpretive validity. In what follows, we discuss these validity threats.

Theoretical validity refers to the extent to which the theoretical explanation fits the data. Inevitably, researchers' biases exist in the selection of studies, in particular in the design of the search string, selection of databases and inclusion/exclusion criteria. However, we minimized this threat by following with rigor the guidelines for systematic mapping in SE. Additionally, the search was carried out independently by two authors. When discrepancies emerged, they were discussed among the authors until consensus was reached. Another threat to theoretical validity is publication bias, which may be probably caused by the novelty of the topic. Acknowledging the novelty of the topic and the lack of empirical evidence, to minimize publication biases and to ensure the reproducibility of our study, we created a reproducibility package, accessible as archived open data [38].

Descriptive validity refers to the degree to which findings are described in an accurate manner. This threat is mainly related to the design of data extraction forms. Data extraction forms (publication type, publication year, authors, title, keywords, publication source, research type, contribution type, research topic, blockchain aspects and platforms) were designed by the first author and reviewed by the other authors, in accordance with the research questions and research scope.

Interpretive validity refers to the extent conclusions are reasonable taking into account the data. The interpretation of the findings was conducted by the first author and validated by the other authors with experience in secondary studies in the SE field. A threat to interpretation validity is the categorization of a limited sample of studies. At first glance, this threat seems to complicate the comprehensive interpretation and discussion of the findings. However, we perceive this as an opportunity for further research in a variety of unexplored SE dimensions and blockchain aspects.

6. Conclusions

Recently, blockchain technology has attracted many organizations, due to its intrinsic potential. The potential of this novel technology has been explored in a variety of domains, such as financial applications, supply chain management and healthcare. However, the use of blockchain in software engineering has not received enough attention. In this regard, we carried out a systematic mapping study to identify software engineering applications enabled by blockchain.

This systematic mapping follows the guidelines for systematic mappings in SE provided by Petersen et al. [28]. After a careful search and selection process, we identified 22 relevant studies and extracted data within three facets: research type, research topic and contribution type. Our findings suggest an increasing trend of studies since 2018 (20 out of 22 studies, 91%). Several studies focused on replacing centralized systems, such as GitHub, Travis CI and cloud-based package managers, while other studies focused on using smart contracts to automate SDLC activities, such as the acceptance phase, payments to software engineers, and compliance adherence. Additionally, blockchain facilitates trusted collaboration and coordination in distributed software development, software provenance, and software integrity assessment.

As the application of blockchain in software engineering is an emerging field, researchers should contribute with more prototypes and proofs-of-concept in order to enhance the understanding of contributions that blockchain can bring to the SE landscape. More research efforts devoted to this topic can encourage practitioners to implement blockchain-oriented SE applications. Based on our analysis we present the following future directions: *(i)* automatically verify software design and requirements against predefined acceptance criteria, by means of smart contracts *(ii)* investigate the holistic impacts of blockchain on software development quality and efficiency *(iii)* investigate the align-

ment between blockchain principles and agile principles *(iv)* compare features of different blockchain platforms and investigate their applicability to SE use cases *(v)* explore the impact of blockchain 4.0 on software engineering, and *(vi)* address the need for professionals competent in both blockchain and software engineering. Our future work will focus on developing and validating a blockchain-enabled framework for requirements management and traceability in interorganizational software projects.

Author Contributions: Conceptualization, S.D., R.C.-P. and M.S.-G.; methodology, S.D.; validation, R.C.-P. and M.S.-G.; formal analysis, S.D., R.C.-P. and M.S.-G.; data curation, S.D. and M.S.-G.; writing—original draft preparation, S.D.; writing—review and editing, R.C.-P. and M.S.-G.; supervision, R.C.-P. All authors have read and agreed to the published version of the manuscript.

Funding: This research and the APC was funded by Østfold University College.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data supporting reported results can be found at https://doi.org/10.6084/m9.figshare.12197928 (accessed on 1 March 2021).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Selected studies.

	Reference	Year	Title	Source
1	[41]	2015	A novel method for decentralized peer-to-peer software license validation using cryptocurrency blockchain technology	Conference (ACSC)
2	[29]	2017	Design of a smart contract based autonomous organization for sustainable software	Conference (e-Science)
3	[44]	2018	ChainSoft: Collaborative software development using smart contracts	Workshop (CRYBLOCK)
4	[45]	2018	Blockchain applications for agile methodologies	Conference (XP)
5	[49]	2018	A study of blockchain based on graph database for software quality measurement integrity	Conference (ICTC)
6	[50]	2018	Powering software sustainability with blockchain	Conference (CASCON)
7	[51]	2018	CoderChain: A blockchain community for coders	Conference (HotICN)
8	[56]	2018	(WIP) Blockhub: Blockchain-based software development system for untrusted environments	Conference (CLOUD)
9	[57]	2018	Smart collaboration mechanism using blockchain technology	Conference (EdgeCom)
10	[37]	2019	Framework for trustworthy software development	Workshop (ASEW)
11	[30]	2019	Blockchain-based software engineering	Conference (ICSE-NIER)
12	[46]	2019	Applying blockchain to improve the integrity of the software development process	Conference (EUROSPI)
13	[47]	2019	Blockchain-based marketplace for software testing	Conference (PST)
14	[52]	2019	Distributed software dependency management using blockchain	Conference (PDP)
15	[53]	2019	CAG: Compliance adherence and governance in software delivery using blockchain	Workshop (WETSEB)
16	[54]	2019	Blinker: A blockchain-enabled framework for software provenance	Conference (APSEC)
17	[55]	2019	ShIFt—Software identity framework for global software delivery	Conference (ICGSE)
18	[42]	2020	Towards efficient and secure global software development using blockchain	Conference (EASE)
19	[43]	2020	Blockchain-oriented requirements engineering: a framework	Conference (RE)
20	[48]	2020	A blockchain-based testing approach for collaborative software development	Conference (Blockchain)
21	[58]	2020	Application of blockchain for trusted collaboration in collaborative software development	Conference (COMPSAC)
22	[59]	2020	Are software engineers incentivized enough? An outcome-based incentive framework using tokens	Workshop (IWBOSE)

References

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. *Cryptogr. Mail. List* **2009**. Available online: https://bitcoin.org/en/bitcoin-paper (accessed on 11 February 2021).
2. Swan, M. *Blockchain: Blueprint for a New Economy*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2015.
3. Ethereum Whitepaper | Ethereum.org. Available online: https://ethereum.org/en/whitepaper/ (accessed on 1 February 2021).
4. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Yellick, J.; Ferris, C.; Enyeart, D.; Laventman, G.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, New York, NY, USA, 23–26 April 2018; pp. 1–15.
5. Agbo, C.C.; Mahmoud, Q.H.; Eklund, J.M. Blockchain technology in healthcare: A systematic review. *Healthcare* **2019**, *7*, 56. [CrossRef] [PubMed]
6. Porru, S.; Pinna, A.; Marchesi, M.; Tonelli, R. Blockchain-oriented software engineering: Challenges and new directions. In Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), Buenos Aires, Argentina, 20–28 May 2017; pp. 169–171.
7. Destefanis, G.; Marchesi, M.; Ortu, M.; Tonelli, R.; Bracciali, A.; Hierons, R. Smart contracts vulnerabilities: A call for blockchain software engineering? In Proceedings of the 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE), Campobasso, Italy, 20 March 2018; pp. 19–25.
8. Marchesi, M. Why blockchain is important for software developers, and why software engineering is important for blockchain software (Keynote). In Proceedings of the 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE), Campobasso, Italy, 20 March 2018; p. 1.
9. Colomo-Palacios, R. Cross fertilization in software engineering. In *Systems, Software and Services Process Improvement*; Yilmaz, M., Niemann, J., Clarke, P., Messnarz, R., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 3–13.
10. IEEE Std 610.12-1990. *IEEE Standard Glossary of Software Engineering Terminology*; IEEE: New York, NY, USA, 2008; pp. 1–84. [CrossRef]
11. Software Engineering Course (SWEBOK) | IEEE Computer Society. Available online: https://www.computer.org/education/bodies-of-knowledge/software-engineering (accessed on 20 February 2021).
12. Ebert, C.; Kuhrmann, M.; Prikladnicki, R. Global software engineering: Evolution and trends. In Proceedings of the 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE), Orange County, CA, USA, 2–5 August 2016; pp. 144–153.
13. Niazi, M.; Mahmood, S.; Alshayeb, M.; Riaz, M.R.; Faisal, K.; Cerpa, N.; Khan, S.U.; Richardson, I. Challenges of project management in global software development: A client-vendor analysis. *Inf. Softw. Technol.* **2016**, *80*, 1–19. [CrossRef]
14. McHugh, O.; Conboy, K.; Lang, M. Agile practices: The impact on trust in software project teams. *IEEE Softw.* **2011**, *29*, 71–76. [CrossRef]
15. Dikert, K.; Paasivaara, M.; Lassenius, C. Challenges and success factors for large-scale agile transformations: A systematic literature review. *J. Syst. Softw.* **2016**, *119*, 87–108. [CrossRef]
16. Shahin, M.; Babar, M.A.; Zhu, L. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access* **2017**, *5*, 3909–3943. [CrossRef]
17. Rempel, P.; Patrick, M.; Kuschke, T.; Philippow, I. Requirements traceability across organizational boundaries—A survey and taxonomy. In Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality, Essen, Germany, 8–11 April 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 125–126.
18. Mao, K.; Capra, L.; Harman, M.; Jia, Y. A survey of the use of crowdsourcing in software engineering. *J. Syst. Softw.* **2017**, *126*, 57–84. [CrossRef]
19. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
20. Tschorsch, F.; Scheuermann, B. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2084–2123. [CrossRef]
21. Sagar Bharadwaj, K.S.; Dharanikota, S.; Honawad, A.; Chandrasekaran, K. *Blockchain Research and Applications: A Systematic Mapping Study*; Patel, D., Nandi, S., Mishra, B.K., Shah, D., Modi, C.N., Shah, K., Bansode, R.S., Eds.; IC-BCT 2019; Springer: Singapore, 2020; pp. 141–159.
22. Niranjanamurthy, M.; Nithya, B.N.; Jagannatha, S. Analysis of Blockchain technology: Pros, cons and SWOT. *Clust. Comput.* **2019**, *22*, 14743–14757. [CrossRef]
23. Marchesi, L.; Marchesi, M.; Destefanis, G.; Barabino, G.; Tigano, D. Design patterns for gas optimization in ethereum. In Proceedings of the 2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), London, ON, Canada, 18 February 2020; pp. 9–15.
24. Vacca, A.; Di Sorbo, A.; Visaggio, C.A.; Canfora, G. A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges. *J. Syst. Softw.* **2021**, *174*, 110891. [CrossRef]
25. Christidis, K.; Devetsikiotis, M. Blockchains and smart contracts for the Internet of Things. *IEEE Access* **2016**, *4*, 2292–2303. [CrossRef]
26. Pinna, A.; Ibba, S.; Baralla, G.; Tonelli, R.; Marchesi, M. A massive analysis of ethereum smart contracts empirical study and code metrics. *IEEE Access* **2019**, *7*, 78194–78213. [CrossRef]

27. Tariq, F.; Colomo-Palacios, R. Use of blockchain smart contracts in software engineering: A systematic mapping. In *Computational Science and Its Applications–ICCSA*; Misra, S., Gervasi, O., Murgante, B., Stankova, E., Korkhov, V., Torre, C., Rocha, A.M.A.C., Taniar, D., Apduhan, B.O., Tarantino, E., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 327–337.
28. Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. Systematic mapping studies in software engineering. In Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), Bari, Italy, 26–27 June 2008; pp. 1–10.
29. Alimoglu, A.; Ozturan, C. Design of a smart contract based autonomous organization for sustainable software. In Proceedings of the 2017 IEEE 13th International Conference on e-Science (e-Science), Auckland, New Zealand, 24–27 October 2017; pp. 471–476.
30. Beller, M.; Hejderup, J. Blockchain-based software engineering. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), Montreal, QC, Canada, 25–31 May 2019; pp. 53–56.
31. Makridakis, S.; Christodoulou, K. Blockchain: Current challenges and future prospects/applications. *Future Internet* **2019**, *11*, 258. [CrossRef]
32. *Kitchenham B Guidelines for performing Systematic Literature Reviews in Software Engineering*; EBSE Technical Report; Keele University and University of Durham: Durham, UK, 2007.
33. Rocha, H.; Ducasse, S. Preliminary steps towards modeling blockchain oriented software. In Proceedings of the 1st International Workshop on Code Hunt Workshop on Educational Software Engineering, Gothenburg, Sweden, 27 May–3 June 2018; pp. 52–57.
34. Al-Jaroodi, J.; Mohamed, N.; Jawhar, I.; Lazarova-Molnar, S. Software engineering issues for cyber-physical systems. In Proceedings of the 2016 IEEE International Conference on Smart Computing (SMARTCOMP), St. Louis, MO, USA, 18–20 May 2016; pp. 1–6.
35. Berger, C.; Penzenstadler, B.; Drögehorn, O. On using blockchains for safety-critical systems. In Proceedings of the 2018 IEEE/ACM 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS), Gothenburg, Sweden, 27 May–3 June 2018; pp. 30–36.
36. Singi, K.; Jagadeesh Chandra Bose, R.P.; Podder, S.; Burden, A.P. Trusted software supply chain. In Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA, 11–15 November 2019; pp. 1212–1213.
37. Jagadeesh Chandra Bose, R.P.; Singi, K.; Kaulgud, V.; Phokela, K.K.; Podder, S. Framework for trustworthy software development. In Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW), San Diego, CA, USA, 11–15 November 2019; pp. 45–48.
38. Software Engineering Applications enabled by Blockchain Technology: A Systematic Mapping Study. Available online: https://figshare.com/articles/dataset/Software_Engineering_Applications_enabled_by_Blockchain_Technology_A_Systematic_Mapping_Study/12197928 (accessed on 11 February 2021).
39. Jalali, S.; Wohlin, C. Systematic literature studies: Database searches vs. backward snowballing. In Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Lund, Sweden, 20–21 September 2012; pp. 29–38.
40. Macrinici, D.; Cartofeanu, C.; Gao, S. Smart contract applications within blockchain technology: A systematic mapping study. *Telemat. Inform.* **2018**, *35*, 2337–2354. [CrossRef]
41. Herbert, J.; Litchfield, A. A novel method for decentralised peer-to-peer software license validation using cryptocurrency blockchain technology. In Proceedings of the 38th Australasian Computer Science Conference, Sydney, Australia, 27–30 January 2015; pp. 27–35.
42. Akbar, M.A.; Al-Sanad, A.; AlSanad, A.A.; Ghmaei, A.; Shafiq, M.; Kamal, T. Towards efficient and secure global software development using blockchain. In Proceedings of the Evaluation and Assessment in Software Engineering, Association for Computing Machinery, Trondheim, Norway, 15–17 April 2020; pp. 493–498.
43. Demi, S. Blockchain-oriented requirements engineering: A framework. In Proceedings of the 2020 IEEE 28th International Requirements Engineering Conference (RE), Zurich, Switzerland, 27 August–4 September 2020; pp. 428–433.
44. Król, M.; Reñé, S.; Ascigil, O.; Psaras, I. Chainsoft: Collaborative software development using smart contracts. In Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, Munich, Germany, 15 June 2018; pp. 1–6.
45. Lenarduzzi, V.; Lunesu, M.I.; Marchesi, M.; Tonelli, R. Blockchain applications for agile methodologies. In Proceedings of the 19th International Conference on Agile Software Development: Companion, Porto, Portugal, 21–25 May 2018; pp. 1–3.
46. Yilmaz, M.; Tasel, S.; Tuzun, E.; Gulec, U.; O'Connor, R.V.; Clarke, P.M. Applying blockchain to improve the integrity of the software development process. In *Advances in Service-Oriented and Cloud Computing*; Metzler, J.B., Ed.; Springer International Publishing: Cham, Switzerland, 2019; pp. 260–271.
47. Wang, Y.; Samavi, R.; Sood, N. Blockchain-based marketplace for software testing. In Proceedings of the 2019 17th International Conference on Privacy, Security and Trust (PST), Fredericton, NB, Canada, 26–28 August 2019; pp. 1–3.
48. Yau, S.S.; Patel, J.S. A blockchain-based testing approach for collaborative software development. In Proceedings of the 2020 IEEE International Conference on Blockchain (Blockchain), Rhodes, Greece, 2–6 November 2020; pp. 98–105.
49. Kim, D.; Kim, H. A study of blockchain based on graph database for software quality measurement integrity. In Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 17–19 October 2018; pp. 1457–1460.

50. Badreddin, O. Powering software sustainability with blockchain. In Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering, Markham, OT, Canada, 29–31 October 2018; pp. 315–322.
51. Lin, Y.; Qi, Z.; Wu, H.; Yang, Z.; Zhang, J.; Wenyin, L. Coderchain: A blockchain community for coders. In Proceedings of the 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN), Shenzhen, China, 15–17 August 2018; pp. 246–247.
52. D'Mello, G.; Gonzalez-Velez, H. Distributed software dependency management using blockchain. In Proceedings of the 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Pavia, Italy, 13–15 February 2019; pp. 132–139.
53. Singi, K.; Kaulgud, V.; Bose, R.J.C.; Podder, S. CAG: Compliance adherence and governance in software delivery using blockchain. In Proceedings of the 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), Montreal, QC, Canada, 27 May 2019; pp. 32–39.
54. Bose, R.J.C.; Phokela, K.K.; Kaulgud, V.; Podder, S. BLINKER: A blockchain-enabled framework for software provenance. In Proceedings of the 2019 26th Asia-Pacific Software Engineering Conference (APSEC), Putrajaya, Malaysia, 2–5 December 2019; pp. 1–8.
55. Singi, K.; Kaulgud, V.; Bose, R.J.C.; Podder, S. ShIFt-Software identity framework for global software delivery. In Proceedings of the 2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE), Montreal, QC, Canada, 25–26 May 2019; pp. 122–128.
56. Ulybyshev, D.; Villarreal-Vasquez, M.; Bhargava, B.; Mani, G.; Seaberg, S.; Conoval, P.; Pike, R.; Kobes, J. (WIP) Blockhub: Blockchain-based software development system for untrusted environments. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018; pp. 582–585.
57. Jhala, K.S.; Oak, R.; Khare, M. Smart collaboration mechanism using blockchain technology. In Proceedings of the 2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), Shanghai, China, 22–24 June 2018; pp. 117–121.
58. Yau, S.S.; Patel, J.S. Application of blockchain for trusted coordination in collaborative software development. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 1036–1040.
59. Singi, K.; Kaulgud, V.; Bose, R.J.C.; Choudhury, S.G.; Podder, S.; Burden, A.P. Are software engineers incentivized enough? An outcome based incentive framework using tokens. In Proceedings of the 2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), London, ON, Canada, 18 February 2020; pp. 37–47.
60. Wang, S.; Ouyang, L.; Yuan, Y.; Ni, X.; Han, X.; Wang, F.-Y. Blockchain-enabled smart contracts: Architecture, applications, and future trends. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 2266–2277. [CrossRef]
61. Laskar, S.; Mishra, D. Qualified vector match and merge algorithm (QVMMA) for DDoS prevention and mitigation. *Procedia Comput. Sci.* **2016**, *79*, 41–52. [CrossRef]
62. Clarke, P.; O'Connor, R.V.; Leavy, B. A complexity theory viewpoint on the software development process and situational context. In Proceedings of the 2016 IEEE/ACM International Conference on Software and System Processes (ICSSP), Austin, TX, USA, 14–15 May 2016; pp. 86–90.
63. Chang, S.E.; Chen, Y. When blockchain meets supply chain: A systematic literature review on current development and potential applications. *IEEE Access* **2020**, *8*, 62478–62494. [CrossRef]
64. Craggs, B.; Rashid, A. Trust beyond computation alone: Human aspects of trust in blockchain technologies. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS), Montreal, QC, Canada, 25–31 May 2019; pp. 21–30.
65. Werbach, K. *The Blockchain and the New Architecture of Trust*; The MIT Press: Cambridge, MA, USA, 2018.
66. Kuo, T.-T.; Rojas, H.Z.; Ohno-Machado, L. Comparison of blockchain platforms: A systematic review and healthcare examples. *J. Am. Med. Inform. Assoc.* **2019**, *26*, 462–478. [CrossRef]
67. Petersen, K.; Vakkalanka, S.; Kuzniarz, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Inf. Softw. Technol.* **2015**, *64*, 1–18. [CrossRef]