A Blockchain-Enabled Framework for Requirements Traceability

Selina Demi, Mary Sánchez-Gordón(✉), and Ricardo Colomo-Palacios

Østfold University College, Halden, Norway
{selina.demi,mary.sanchez-gordon,
ricardo.colomo-palacios}@hiof.no

Abstract. Requirements traceability has been broadly recognized by researchers as an important quality of any software development process. However, among stakeholders, requirements traceability is often perceived as an extra task that disrupts their workflow. This perceived overhead demotivates stakeholders to participate in the creation, maintenance and use of traceability links. The challenges of implementing requirements traceability are amplified when complex and large-scale software systems are developed by cross-organizational and distributed teams. Different organizational backgrounds, conflicting objectives, and organizational boundaries lead to trust issues that complicate the implementation of traceability in such settings. In this paper, the authors propose a blockchain-enabled framework for requirements traceability. This framework aims to: (i) enable a holistic and reliable view of artifacts and traceability links, (ii) provide an incentive mechanism for creators of traceability links, (iii) ensure the authenticity and quality of traceability links by means of voting mechanisms, (iv) facilitate comprehension from traceability information through query services, and (v) enable interactive graphical visualization of traceability links.

Keywords: Requirements traceability · Blockchain technology · Smart contracts · Distributed software development

1 Introduction

Requirements traceability refers to "the ability to follow the life of a requirement in both a forward and backward direction" [1]. The need to maintain bidirectional traceability of requirements for quality purposes has been formulated by software process improvement models, such as CMMI, ISO/IEC 15504 (SPICE) [2] and more recently ISO/IEC 33000. Requirements traceability contributes to the development of high-quality software, as it supports many activities of the software development lifecycle, for instance, software maintenance [3], project management [4], change management and impact analysis [5]. However, in practice, traceability is perceived as an extra task that practitioners need to perform or as an activity that disrupts their workflow [6]. This perceived overhead can be explained by the provider-user gap, meaning that creators of trace links are often not the ones who use them [6, 7]. For instance, developers create links between implementation

© Springer Nature Switzerland AG 2021

tasks and source code commits which are actually used by project managers to track project's progress. Thereby, developers become demotivated and set a low priority to traceability tasks, which may lead to incorrect or missing links. Maro et al. [6] proposed the incorporation of gamification elements into traceability tools to increase the motivation and engagement of developers in traceability creation and maintenance tasks. Moreover, Wohlrab et al. [7] carried out 24 interviews to explore the relation between traceability management and collaboration. The respondents proposed the inclusion of voting features into traceability tools to enable stakeholders to jointly indicate incorrect links or to agree with the connected artifacts. This collaborative effort improves the effort/benefit ratio, which in turn motivates developers to contribute to traceability management.

As the complexity of software increases, the development is carried out by cross-organizational and distributed teams. This paradigm complicates traceability, particularly when distributed teams need to share artifacts [6]. In such environments, previous literature proposed the use of a centralized data storage in which all artifacts are stored and accessed by distributed stakeholders [6]. However, artifacts provided by distributed teams cannot always be trusted, as they may have malicious intentions [8]. The participation of third-party vendors exacerbates trust issues, due to different organizational background, conflicting objectives, and organizational boundaries [9]. Different organizations may use different tools, methodologies, and processes that reside within the organizational boundaries, making it difficult to leverage requirements traceability in an efficient manner [9]. Additionally, organizational objectives of one organization may contradict objectives of the other organizations that are involved in the project. For instance, organizations may create incompatible links in terms of type or granularity, which leads to unusable trace links [6]. Finally, organizational boundaries can imply restricted access to some artifacts due to confidentiality constraints [6, 9] which in turn complicates the creation of complete traceability. To address these challenges, there is a need for a reliable and shared traceability knowledge base, in order to keep track of all artifacts and trace links created by distributed stakeholders throughout the software development lifecycle (SDLC). This can be achieved by means of blockchain technology.

Blockchain is a distributed ledger that stores transaction records in blocks. Each block includes the hash of the preceding block to point to the previously validated block in the chain [10]. This structure of a cryptographically linked list ensures immutability which refers to the inability to tamper with the contents stored on the blockchain. The main utility of blockchain is that it enables the exchange of data or transactions among untrusted participants in a distributed network, without relying on a centralized trusted party. Centralized third parties are prone to failures, malfunctions, and security compromises which may lead to system unavailability [11]. Blockchain-based systems overcome these risks, as every participant keeps a copy of the ledger and can verify the legitimacy of the transactions [10]. Since 2015, the potential of blockchain technology extended greatly due to the introduction of smart contracts [12]. Smart contracts are self-executing computer programs that run across the blockchain network and enable trusted transactions and agreements among different parties [13]. The results of smart contracts execution are verified by the nodes of the network and stored on the distributed ledger. Smart contracts were originally conceived to automatically implement the terms of the contract that two parties agreed upon in a trustless environment [14]. Nowadays, the scope of smart contracts has been largely extended to perform any conceivable task, similarly to general-purpose software programs [14]. Smart contracts are intended for a variety of application domains, ranging from financial, notary, game, wallet to library which comprises general-purpose operations that can be used by other smart contracts [15].

In this paper, we propose a blockchain-enabled framework for requirements traceability. To the best of our knowledge, this is the first study that uses blockchain to keep track of artifacts and traceability links in distributed settings. This framework aims to:

- enable a holistic and reliable view of artifacts and traceability links
- provide an incentive mechanism for creators of traceability links
- ensure the authenticity and quality of traceability links by means of voting mechanisms
- facilitate comprehension from traceability information through query services
- enable interactive graphical visualization of traceability links

The paper is structured as follows: Related work is presented in Sect. 2. Section 3 describes the proposed framework, and Sect. 4 concludes the work and presents future research directions.

2 Related Work

Mader et al. [16] emphasized the importance of a traceability information model in facilitating traceability creation and maintenance. These authors presented the traceability information model used by their prototype, namely, traceMaintainer. They conducted experiments with subjects who were provided with the traceability information model and were required to create trace links between use cases and analysis classes and trace links between analysis classes and design classes. The subjects reported that they were satisfied with the guidance provided when creating trace links and that traceMaintainer prevented them from creating trace links in an inappropriate manner. The authors advocate the use of traceability information models as they enable different analyses, such as validating traces, impact analysis and change propagation, coverage analysis, and relation count analysis.

Cleland-Huang et al. [17] proposed a model-based approach that enables stakeholders to plan and execute traceability strategies in a graphical modeling environment. This approach consists of four layers: strategic layer, document management layer, stored query layer, and executable layer. The strategic layer represents the traceability graph structure which consists of artifacts and trace paths, using a standard XML format. The document management layer documents individual set of artifacts using a standard XML representation. The stored query layer constructs queries for primitive traces between adjacent artifacts types, and composite traceability paths between non-adjacent artifacts in the strategic traceability graph. Finally, the executable layer provides the user interface to display the pre-defined trace queries and visualize the results. Our study adopts these conceptual layers, albeit it implements them in a different manner.

Elamin and Osman [5] proposed a user-defined traceability metamodel that uses XML patterns to define artifacts, trace links, and trace type rules. According to these authors, the main limitations of traceability approaches lie in the representation and storage of traceability information. To address these limitations, the authors implemented a graph traceability repository to store artifacts and trace links. The benefits of the graph repository were demonstrated by applying it to three varying datasets. The results confirmed that the graph repository outperforms traceability matrices, cross-reference tables and relational databases in terms of visualizing trace links, representing multi-dimensional relations and performance. We adopt the concept of rules for trace links semantics, however we encode these rules into smart contracts and store artifacts and trace links on a distributed ledger.

None of the aforementioned approaches address requirements traceability in distributed environments. In this regard, Rempel et al. [9] investigated the need for requirements traceability in inter-organizational software projects. The authors carried out semi-structured interviews with 17 organizations. The results indicated that on the one hand requirements traceability has the potential to address inherent issues of inter-organizational software projects, such as compliance, operational excellence and communication between parties. On the other hand, the different organizational backgrounds, conflicting objectives, and organizational boundaries were found to complicate the application of requirements traceability. The authors presented the following guidelines for distributed requirements traceability: ensure the reliability and availability of traceability information, mitigate conflicting objectives and bridge the technological gap that exists between clients and suppliers. Despite the valuable insights, this study does not provide further information on how these guidelines can be implemented.

Furthermore, recent literature has encouraged the cross-fertilization of software engineering and hyped technologies, such as blockchain technology [18, 19]. In fact, we have observed an increasing trend of studies that use blockchain to support software engineering activities during the last three years [20]. In what follows, we present a few of these studies: Yilmaz et al. [21] used blockchain to improve the integrity of the software development process. The authors proposed an incentive mechanism where developers compete for developing the best code, instead of being assigned a specific task by the project manager. This proposal may be valuable to address trust issues, particularly in large-scale agile development. Bose et al. [22] proposed a blockchain-oriented framework for trustworthy software provenance in global software development. The framework captures provenance data from the variety of tools used throughout the SDLC and transforms them according to PROV-specifications. The authenticity of these data is verified by authorized personnel by means of voting mechanisms. Recently, Singi et al. [23] proposed an incentive framework enabled by blockchain technology that captures events throughout the entire software development lifecycle, analyzes these events for their compliance to incentive policies by means of smart contracts, and automatically delivers incentives in the form of digitized tokens to software engineers, accordingly. These studies provide valuable insights into the applications of blockchain in the software engineering landscape, however none of the aforementioned studies is devoted to the use of blockchain for requirements traceability in distributed settings.

3 Proposed Framework

Figure 1 depicts the blockchain-enabled requirements traceability framework. The proposed framework consists of the following components:

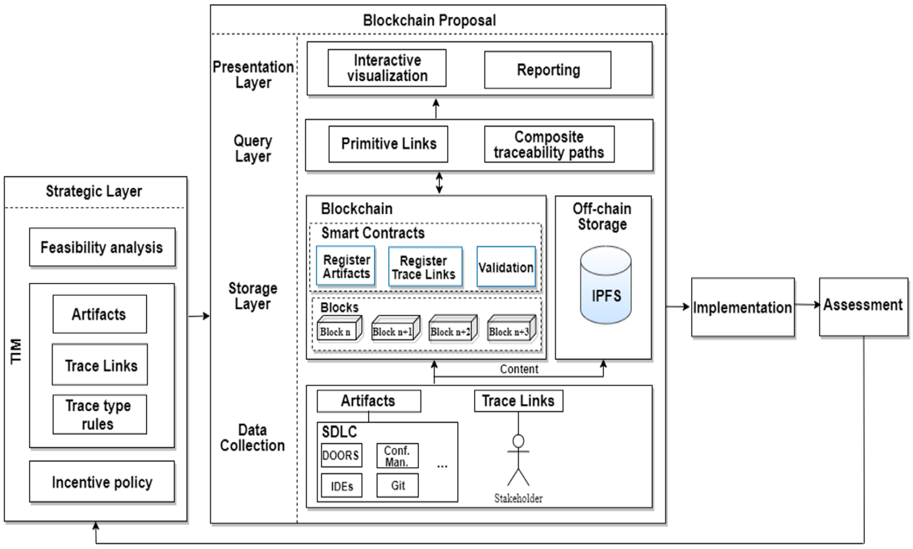

Fig. 1. Blockchain-enabled requirements traceability framework

Strategic Layer. A feasibility analysis should be carried out in order to investigate whether the application of blockchain for requirements traceability is required and feasible in the given environment. This analysis should take into consideration the alignment between blockchain features and requirements traceability strategies, alternative tools, and technologies for representing and storing traceability links. The latter has been achieved in the form of traceability matrices, cross-reference tables, relational databases, and graph traceability repositories [5]. It is also vital that the analysis explores the benefits of applying blockchain for requirements traceability, along with challenges, such as implementation costs, technical, regulatory and governance challenges. The next component of the strategic layer is the traceability information model (TIM) which provides guidance on what software artifacts to trace and what relations to establish, and consequently prevents inconsistent results in large projects with many stakeholders [16]. Determining the traceable artifacts and relations in advance has been considered a best practice for establishing the traceability environment [24]. Figure 2 depicts a simple traceability information model which consists of three main elements, as follows [5]:

- *Artifacts* to define what artifacts should be traced and their properties.
- *Trace links* to define relations between artifacts based on source artifact ID and destination artifact ID.

- *Trace type rules* to define the naming rules for the trace links. For instance, if the source code is related to the requirement, then the name of the relation is "satisfy".

These elements can be encoded into smart contracts to enforce the registration of only those artifacts and trace links that are needed in the project and automatically identify trace links semantics.

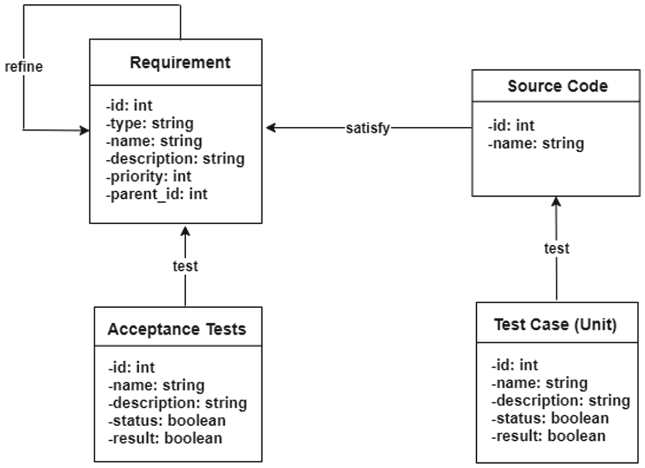

Fig. 2. A simple traceability information model

The last component of the strategic layer is the incentive policy that aims to define who is eligible to create trace links, how to validate the quality of trace links, e.g., can be validated by stakeholders using traceability quality metrics such as correctness, timeliness, accuracy, completeness, consistency and usefulness [25], and how much incentive goes for the creation of trace links based on their priority.

Blockchain Proposal. The blockchain proposal forms the core component of the framework and consists of:

- *Data collection.* A variety of tools are used throughout the software development lifecycle, such as Rational DOORS for requirements management, and Git as a version control system. These tools generate artifacts that are defined in the traceability information model. Artifacts generated from these disparate sources can be captured automatically by means of data ingestion tools/plugins [22] and are parsed prior to being recorded on the blockchain. Additionally, stakeholders create trace links manually and invoke the respective smart contract to register these links on the distributed ledger.
- *Storage layer and smart contracts.* Smart contracts are created to enable the following functions: register artifacts, e.g., requirements (id, type, name, description, priority, parent_id), register trace links (source_artifact_id, destination_artifact_id, trace_type), validate trace links quality and reward trace links creators, accordingly.

On platforms such as Ethereum, the storage costs gas and can lead to network synchronization issues and the high consumption of disk space on the nodes [26]. To resolve storage limitations, the framework allows to store artifacts' contents off-chain in immutable storage, such as IPFS (InterPlanetary File System). Moreover, the quality of trace links is validated by approvers or verifiers whose selection depends on voting policies. The approvers provide their consent or rejection which is logged as a vote on the distributed ledger. The smart contract calculates cumulative votes and accepts or rejects trace links based on the vote score. It is noteworthy that the assessment of traceability links quality is challenging because it requires manual checking [6]. Finally, the incentive policies are encoded in the smart contract that distributes digitized tokens to stakeholders who create quality trace links.
- *Query layer.* This layer facilitates comprehension from traceability information stored on the blockchain. Traceability-related queries must be constructed from primitive and composite links [17]. Primitive links are links between adjacent artifacts types defined in the traceability information model. These queries comprise simple forms of traces, for instance, return the list of requirements satisfied by source code or using filters to include only artifacts with a specific attribute value. Composite trace links are more complicated to implement as they take place between non-adjacent artifacts, for instance return requirements that trace to source code which failed its test case [17].
- *Presentation layer.* This layer is responsible for the visualization of traceability-related information. Traceability visualization enhances stakeholders' ability to comprehend relationships between artifacts. Previous literature reports on difficulties in uncovering insights from traceability information due to the fact that trace links are represented through lists or mega tables [6, 27]. Other approaches use two-dimensional graphical formats such as hierarchical leaf node and tree view. These representations fail to explore relations between different artifacts in an interactive manner which aids in comprehending the overall system [28]. Our framework suggests hierarchical and interactive visualization of trace links to enhance traceability comprehension.

Implementation. In this phase, the blockchain proposal is developed and implemented. The development of the blockchain proposal requires close cooperation between blockchain developers and requirements traceability experts or professionals with skills in both blockchain and requirements traceability. The need for new professional roles has been also observed in the broader field of blockchain-oriented software engineering by Porru et al. [29]. Furthermore, the best fitting blockchain platform should be selected by mapping the requirements of the desired system and blockchain features. As the number of blockchain platforms is growing rapidly, the selection of the appropriate platform becomes challenging. To guide the selection process, we refer to previous literature that provides a grounded blockchain platform selection process [30]. The main items to consider comprise network acessibility, smart contract support, and whether tokens are required or not [30]. For instance, open-source software with diverse contributors might require public accessibility whereas the development of complex and large-scale software among a set of known distributed teams or organizations might require restricted accessibility. Finally, disruptive technologies such as blockchain are adopted once organizational resistance is overcome which in turn can be achieved if the organization perceives the value of such a system [31]. Therefore, it is important to communicate the value of implementing blockchain for requirements traceability in a transparent and clear manner to all the participants involved in the software development lifecycle.

Assessment. In this phase, the contributions of the blockchain proposal to the software development lifecycle are assessed. The proposed framework provides an incentive mechanism to create quality trace links, which motivates stakeholders to participate in traceability creation and maintenance. Furthermore, the framework enables a holistic and trustworthy view of artifacts and trace links and presents them in a hierarchical and graphical way which encourages the use of traceability to support SDLC activities. In turn, the increased use of traceability improves the performance of practitioners in solving SDLC tasks, for instance maintenance tasks. The performance can be measured as the combination of the time to solve the task and correctness of the solution [3]. Finally, the increased quality and completeness of traceability has a positive impact on software quality which can be measured in terms of defect rate [32]. A reduced defect rate implies less need for software maintenance and consequently cost savings that can be quantified [32]. It is worthy to note that the components of the proposed framework are designed to be extensible and should be tailored to organizational requirements. For instance, organizations developing safety-critical systems may use smart contracts to automatically validate compliance to regulations or standards that impose traceability requirements, e.g., ISO 26262 and ASPICE in the automotive domain [6].

4 Conclusion

In this paper, we propose a novel blockchain-enabled framework for requirements traceability. This framework uses blockchain as the backbone of the software development lifecycle, to enable an auditable trail of artifacts and trace links created by multiple distributed stakeholders. Due to its inherent properties, blockchain ensures visibility regarding what/how/when trace links were created and who created them. Additionally, the framework can be used to query and represent traceability-related information to enhance the understanding of the overall system.

Blockchain, as any other technology, does not fit all the use cases in requirements traceability however this ongoing study is exploring a way of investing, using and taking the best from blockchain. Although there is large setup and storage overhead when implementing blockchain for software engineering [33], blockchain can ensure visibility, transparency, traceability and trustworthiness of artifacts and trace links. The framework comprises customizable incentive and voting policies to ensure the trustworthiness of trace links and presents them in an interactive manner to enhance comprehension. These components can potentially encourage the use of traceability to support SDLC activities, and in turn improve the performance of practitioners in solving SDLC tasks and enhance software quality. However, it could also be interesting to improve the proposed framework by incorporating gamification elements in order to enhance the motivation and engagement of practitioners in traceability tasks.

Despite the aforementioned potential benefits, the framework also has limitations that are mainly related to the manual work involved in the creation of smart contracts and in the validation of quality trace links. Another open issue to be addressed is how to resolve conflicts that may occur when different participants claim to have created the same trace links. Furthermore, it is noteworthy that the implementation of blockchain technology for requirements traceability may be challenging due to the limited research efforts in this dimension, the nascent stage of blockchain development and open technical challenges. These limitations have also been identified when implementing blockchain in conventional domains such as supply chain [34]. Although the proposed framework aims to be easy to used, it requires software practitioners to have knowledge of both fields: blockchain technologies and requirements traceability. Further research efforts devoted to the development of prototypes and proofs-of-concept in this area may encourage software development organizations to implement blockchain for requirements traceability. Therefore, the blockchain-enabled framework proposed in this study will be validated by means of blockchain and requirements traceability experts. Then, a blockchain-enabled requirements traceability prototype will be developed and use cases will be performed to test the concept. In so doing, some questions arise: What will be the main benefits of an organization to implement your framework? How easy is the framework for use? What type of knowledge that engineering needs to be able to use the framework? In a common project how should use your framework?

References

1. Gotel, O.C.Z., Finkelstein, C.W.: An analysis of the requirements traceability problem. In: Proceedings of IEEE International Conference on Requirements Engineering, pp. 94–101 (1994)
2. Gotel, O., Cleland-Huang, J., Hayes, J.H., et al.: The quest for ubiquity: a roadmap for software and systems traceability research. In: 2012 20th IEEE International Requirements Engineering Conference (RE), pp. 71–80 (2012)
3. Mäder, P., Egyed, A.: Do developers benefit from requirements traceability when evolving and maintaining a software system? Empir. Softw. Eng. **20**(2), 413–441 (2014). https://doi.org/10.1007/s10664-014-9314-z
4. Murugappan, S., Prabha, D.: Requirement traceability for software development lifecycle. Int. J. Sci. Eng. Res. **8**, 1–11 (2017)
5. Elamin, R., Osman, R.: Implementing traceability repositories as graph databases for software quality improvement. In: 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS), pp. 269–276 (2018)
6. Maro, S., Steghöfer, J.-P., Staron, M.: Software traceability in the automotive domain: challenges and solutions. J. Syst. Softw. **141**, 85–110 (2018). https://doi.org/10.1016/j.jss.2018.03.060
7. Wohlrab, R., Knauss, E., Steghöfer, J.-P., Maro, S., Anjorin, A., Pelliccione, P.: Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture. Requir. Eng. **25**(1), 21–45 (2018). https://doi.org/10.1007/s00766-018-0306-1
8. Yau, S.S., Patel, J.S.: Application of blockchain for trusted coordination in collaborative software development. In: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 1036–1040 (2020)

9. Rempel, P., Mäder, P., Kuschke, T., Philippow, I.: Requirements traceability across organizational boundaries - a survey and taxonomy. In: Doerr, J., Opdahl, A.L. (eds.) REFSQ 2013. LNCS, vol. 7830, pp. 125–140. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37422-7_10
10. Tschorsch, F., Scheuermann, B.: Bitcoin and beyond: a technical survey on decentralized digital currencies. IEEE Commun. Surv. Tutor. **18**, 2084–2123 (2016)
11. Agbo, C.C., Mahmoud, Q.H., Eklund, J.M.: Blockchain technology in healthcare: a systematic review. Healthcare **7**, 56 (2019). https://doi.org/10.3390/healthcare7020056
12. Marchesi, L., Marchesi, M., Destefanis, G., et al.: Design patterns for gas optimization in ethereum. In: 2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), pp. 9–15 (2020)
13. 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. **174**, 110891 (2021). https://doi.org/10.1016/j.jss.2020.110891
14. Pinna, A., Ibba, S., Baralla, G., et al.: A massive analysis of ethereum smart contracts empirical study and code metrics. IEEE Access **7**, 78194–78213 (2019). https://doi.org/10.1109/ACCESS.2019.2921936
15. Bartoletti, M., Pompianu, L.: An empirical analysis of smart contracts: platforms, applications, and design patterns. In: Brenner, M., et al. (eds.) FC 2017. LNCS, vol. 10323, pp. 494–509. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_31
16. Mader, P., Gotel, O., Philippow, I.: Getting back to basics: promoting the use of a traceability information model in practice. In: 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, pp. 21–25 (2009)
17. Cleland-Huang, J., Hayes, J.H., Domel, J.M.: Model-based traceability. In: 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, pp. 6–10 (2009)
18. Marchesi, M.: Why blockchain is important for software developers, and why software engineering is important for blockchain software (Keynote). In: 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE), p. 1 (2018)
19. Colomo-Palacios, R.: Cross fertilization in software engineering. In: Yilmaz, M., Niemann, J., Clarke, P., Messnarz, R. (eds.) EuroSPI 2020. CCIS, vol. 1251, pp. 3–13. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56441-4_1
20. Demi, S., Colomo-Palacios, R., Sánchez-Gordón, M.: Software engineering applications enabled by blockchain technology: a systematic mapping study. Appl. Sci. **11**, 2960 (2021). https://doi.org/10.3390/app11072960
21. 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: Walker, A., O'Connor, R.V., Messnarz, R. (eds.) EuroSPI 2019. CCIS, vol. 1060, pp. 260–271. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-28005-5_20
22. Bose, R.P.J.C., Phokela, K.K., Kaulgud, V., Podder, S.: BLINKER: a blockchain-enabled framework for software provenance. In: 2019 26th Asia-Pacific Software Engineering Conference (APSEC), pp. 1–8 (2019)
23. Singi, K., Kaulgud, V., Chandra Bose, R.P.J., et al.: Are software engineers incentivized enough? An outcome based incentive framework using tokens. In: 2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), pp. 37–47 (2020)
24. Cleland-Huang, J., Berenbach, B., Clark, S., et al.: Best practices for automated traceability. Computer **40**, 27–35 (2007). https://doi.org/10.1109/MC.2007.195
25. Gotel, O., Cleland-Huang, J., Hayes, J.H., et al.: Traceability fundamentals. In: Cleland-Huang, J., Gotel, O., Zisman, A. (eds.) Software and Systems Traceability, pp. 3–22. Springer, London (2012). https://doi.org/10.1007/978-1-4471-2239-5_1

26. Singi, K., Kaulgud, V., Bose, R.P.J.C., Podder, S.: CAG: compliance adherence and governance in software delivery using blockchain. In: 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), pp. 32–39 (2019)
27. Mäder, P., Jones, P.L., Zhang, Y., Cleland-Huang, J.: Strategic traceability for safety-critical projects. IEEE Softw. **30**, 58–66 (2013). https://doi.org/10.1109/MS.2013.60
28. Aung, T.W.W., Huo, H., Sui, Y.: Interactive traceability links visualization using hierarchical trace map. In: 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 367–369 (2019)
29. Porru, S., Pinna, A., Marchesi, M., Tonelli, R.: Blockchain-oriented software engineering: challenges and new directions. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), pp. 169–171 (2017)
30. Farshidi, S., Jansen, S., España, S., Verkleij, J.: Decision support for blockchain platform selection: three industry case studies. IEEE Trans. Eng. Manag. **67**, 1109–1128 (2020)
31. Beck, R., Müller-Bloch, C.: Blockchain as radical innovation: a framework for engaging with distributed ledgers as incumbent organization. In: Proceedings of the 50th Hawaii International Conference on System Sciences (2017)
32. Rempel, P., Mäder, P.: Preventing defects: the impact of requirements traceability completeness on software quality. IEEE Trans. Softw. Eng. **43**, 777–797 (2016)
33. Yau, S.S., Patel, J.S.: A blockchain-based testing approach for collaborative software development. In: 2020 IEEE International Conference on Blockchain (Blockchain), pp. 98–105 (2020)
34. Chang, S.E., Chen, Y.: When blockchain meets supply chain: a systematic literature review on current development and potential applications. IEEE Access **8**, 62478–62494 (2020). https://doi.org/10.1109/ACCESS.2020.2983601