

UNIVERSITY OF OSLO

Selina Demi

Blockchain-oriented Requirements Engineering: A Framework

Thesis submitted for the degree of Philosophiae Doctor

Department of Informatics
Faculty of Mathematics and Natural Sciences

Østfold University College



2023

© Selina Demi, 2023

*Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
No. 2646*

ISSN 1501-7710

All rights reserved. No part of this publication may be
reproduced or transmitted, in any form or by any means, without permission.

Cover: UiO.
Print production: Graphic center, University of Oslo.

To the memory of my grandfather who always believed in me!

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of *Philosophiae Doctor* at the University of Oslo, Faculty of Mathematics and Natural Sciences, Department of Informatics. The research presented here was conducted at the University of Oslo and Østfold University College, under the supervision of Professor Ricardo Colomo-Palacios and Professor Roman Vitenberg. This work was funded by Østfold University College.

• **Selina Demi**

Oslo, September, 2023

Acknowledgment

“Gratitude is not only the greatest of virtues, but the parent of all the others”.
(Cicero)

This PhD dissertation represents the ultimate achievement of a long, yet exciting academic journey. I feel very lucky and grateful for all the possibilities I had throughout my education, that allowed me to accomplish some of my dreams. I would like to express my gratitude to all the knowledgeable and supportive academics, professors and researchers I met at University of Tirana, Norwegian University of Science and Technology (NTNU), Lund University, Østfold University College (HIOF) and University of Oslo (UiO). I had the chance to be part of international student and research communities, live in 6 different cities, integrate in the almost-perfect Scandinavian society, and engage in interesting discussions that always provided me with new perspectives. This is so fascinating for a curious person like me!

This dissertation would not have been possible without the support of Østfold University College. First and foremost, I would like to express my immense gratitude to two very important people in this journey: my main supervisor, Prof. Ricardo Colomo-Palacios and Assoc. Prof. Harald Holone, Dean at Faculty of Computer Science, Engineering and Economics, Østfold University College. Thank you for giving me this great opportunity and I hope I fulfilled your expectations. Thank you, Harald Holone and Monica Kristiansen Holone, for being so positive, motivating and helpful, since the first day we met. Thank you, Ricardo, for guiding me during these three intense years. Your guidance, great energy, and extensive academic experience helped me understand the fundamentals of scientific writing, review process and how to manage and complete a PhD project in a timely fashion. Our endless and constructive discussions regarding research, but not limited to, made me always reflect on how to become a better researcher and a better person. Finally, I would like to thank you for your encouraging words. I remember my first conference presentation and your feedback: *“Very very good and awesome in a word. Very very proud of you”*- 13th July 2020. This message and many others boost my morale and motivated me throughout my intense PhD work.

Another very important person who contributed significantly to enhancing the quality of all the papers published during these three years is Assoc. Prof. Mary Sanchez-Gordon. I have never met a more passionate and hard-working individual than Mary. I am sure that you are going to transmit your passion to all the master students at HIOF. Indeed, you did transmit all your knowledge and positive energy to me! Thank you for having the patience to review my papers and suggest always very good ideas to enhance the quality of my research.

Acknowledgment

I am very grateful for our intense and productive collaboration. I also want to thank a very special person for me at HIOF, Valbona. Dear Valbona, thank you for encouraging me and never leaving me alone when I lived in Halden. Your help and presence during the Covid period played an important role in overcoming a very stressful period in my life.

Many thanks go to my second supervisor, Prof. Roman Vitenberg for introducing me to the UiO environment and contributing to enhancing my knowledge on blockchain technology. Thank you for always being available to answer any questions about my PhD work and also administrative questions regarding the PhD program at UiO. Your high-quality teaching and research work was very motivating for me. It was also great meeting so many talented PhD students at UiO, in particular I am grateful for my good friends: Andrea Merlina, Salar Arbabi and Lea Belosa. I have also met many PhD students and very good friends at the same time at HIOF, in particular I would like to mention: Klaudia Carcani, Xhesika Ramaj and Alireza Khakpour. It is always a pleasure talking with you and I am very grateful for all the conversations, opinions, ideas and support that we shared. Finally, I had the chance to spend a short period of time at NTNU where I had the pleasure to meet many skilled PhD students. In particular, I would like to thank Nabin Chowdhury and Aida Akbarzadeh for being so welcoming, positive and good friends.

During my PhD program, I had the pleasure to collaborate with ByEvolution Factory, an innovative company located in Malaga, Spain. There, I met some of the most knowledgeable and passionate blockchain experts. In particular, I want to express my gratitude to Carlos Velasco and Ramon Cano for hosting and welcoming me in Malaga. Thank you Carlos, for transmitting your impressive passion about technology and innovation. Your contribution was essential for the successful finalization of my PhD thesis.

My warmest thanks go to one of the most important people for me during these three years, Sam. Thank you for always being with me, for supporting me during my most difficult and stressful moments and helping me in everything. I really appreciate having the deepest conversations with you about any kind of topic that one can imagine. Many thanks to my best friend, Geri for being such a helpful, intelligent and positive friend. I will always appreciate your contribution during the last year of my PhD program. I would also like to thank my friend, Enxhi for being such a good and positive person and for encouraging me during the last two years. I am so lucky to have all of you in my life!

Finally, I would like to express my gratitude and immense love to the strong women of my life: my grandmother, my mother and my aunt. This dissertation is not only for me, but also for you and for all the dreams you never had the chance to fulfill. Ju dua shumë dhe shpresoj t'ju kem bërë krenare. Ju jeni forca dhe motivimi im!

Abstract

The paradigm shift from co-located development to global software engineering exacerbates communication, coordination and trust issues among distributed stakeholders that collaborate for the development of complex and large-scale software systems. The participation of contractors or suppliers augments these trust issues, as these entities can have malicious intentions. Suppliers may tamper with software artifacts, such as clients' requirements, in an illegitimate manner, while clients may claim that the software did not meet their requirements and issue bugs even if the software functions in accordance with their requirements specifications. To avoid costly disputes, it is necessary to prove software correctness by tracing requirements to implementation and verification artifacts.

While requirements traceability is an important quality attribute of any software development process, in practice software engineers often assign low priority to traceability tasks, due to the pressure to deliver quality software in a timely fashion. Implementing requirements traceability practices is even more challenging in interorganizational software projects, due to different organizational backgrounds, conflicting objectives, and organizational boundaries. Cross-organizational teams need to share software artifacts and have a holistic view of the software development lifecycle. The traditional approach of storing software artifacts in centralized repositories and guarding their access by means of an access control system is not sufficient in complex and cross-organizational software projects, as it cannot ensure the immutability, integrity, and availability of software artifacts.

This thesis challenges traditional centralized approaches by investigating, designing, implementing, and evaluating decentralized, yet trustworthy solutions for requirements traceability in interorganizational software projects. This thesis proposed blockchain technology, due to its inherent properties, to function as the backbone of the software development lifecycle and store software artifacts generated by a variety of tools, related artifacts, and changes, in a decentralized, yet reliable manner. Leveraging blockchain to create tamper-proof record of requirements, their changes, and related software artifacts can be useful to all the stakeholders of the software lifecycle, including software engineers, project managers, customers, and auditors. These distributed stakeholders are provided with a trustworthy view of what/when/how and by whom software artifacts were created and/or modified and are ensured to work on the same reality. Ultimately, the increased visibility and transparency can enhance communication, coordination, and trust between stakeholders.

To address the topic, this thesis adopted a design science methodology that entailed two core phases: knowledge base and build-evaluate. In the first phase, research foundations were built within the field of software engineering,

requirements traceability and blockchain technology. The PhD student carried out a systematic literature review on 70 recent studies to identify and classify requirements traceability challenges. In addition, a systematic mapping study was conducted to unveil blockchain use cases in software engineering and benefits that blockchain can bring to the software domain. This research knowledge was used in the second phase to design the initial blockchain-enabled framework for requirements traceability that leveraged conventional blockchain platforms and the concept of smart contracts. The proposed framework was refined by performing an interview-based study with blockchain experts and analysing the interview data by means of grounded theory techniques. To address performance efficiency and scalability limitations of conventional blockchain platforms, a prototype based on a neural distributed ledger platform was developed and evaluated by means of software engineering experts' judgement.

This thesis contributes to enhancing the existing knowledge on the blockchain-oriented software engineering field and presenting the first proposal that leverages blockchain technology for trustworthy requirements traceability in interorganizational software projects. The development of the prototype and empirical evidence acquired from interviews with blockchain experts and software engineering experts may encourage researchers and practitioners to develop more and better prototypes and proof-of-concepts to investigate blockchain-enabled requirements traceability use cases. Future research efforts can be devoted to integrating the prototype with existing tools used throughout the software development lifecycle to automate the registration of software artifacts and their changes.

List of Papers

This thesis is a collection of 7 research papers that are listed below.

Paper I (P1)

S. Demi "*Blockchain-oriented Requirements Engineering: A Framework,*" in 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 428–433, IEEE, 2020.

Paper II (P2)

S. Demi, M. Sanchez-Gordon, and R. Colomo-Palacios, "*What have we learnt from the challenges of (semi-) automated requirements traceability? A discussion on blockchain applicability,*" IET Software, vol. 15, no. 6, pp. 391–411, 2021.

Paper III (P3)

S. Demi, R. Colomo-Palacios, and M. Sanchez-Gordon, "*Software Engineering Applications Enabled by Blockchain Technology: A Systematic Mapping Study,*" Applied sciences, vol. 11, no. 7, p. 2960, 2021.

Paper IV (P4)

S. Demi, M. Sánchez-Gordón, and R. Colomo-Palacios, "*A Blockchain-Enabled Framework for Requirements Traceability,*" in Systems, Software and Services Process Improvement: 28th European Conference, EuroSPI 2021, Krems, Austria, September 1–3, 2021, Proceedings, pp. 3–13, Springer, 2021.

Paper V (P5)

S. Demi, M. Sánchez-Gordón, and M. Kristiansen, "*Blockchain for Requirements Traceability: A Qualitative Approach,*" Journal of Software: Evolution and Process, p. e2493, 2022.

Paper VI (P6)

S. Demi, R. Colomo-Palacios, M. Sánchez-Gordón, C. Velasco, and R. Cano, "*A Neural Blockchain for Requirements Traceability: BC4RT Prototype,*" , in Systems, Software and Services Process Improvement: 29th European Conference, EuroSPI 2022, Salzburg, Austria, August 31–September 2, 2022, Proceedings, pp. 45–59, Springer, 2022.

Paper VII (P7)

S. Demi, M. Sánchez-Gordón, and R. Colomo-Palacios, "*Trustworthy and Collaborative Requirements Traceability: Validation of a blockchain-enabled framework,*", Submitted to Journal of Software: Evolution and Process.

Acronyms

AAMI	Association for the Advancement of Medical Instrumentation
AI	Artificial Intelligence
ANSI	American National Standards Institute
ASPICE	Automotive Software Process Improvement Capability dEtermination
BC4RT	Blockchain for Requirements Traceability
BOSE	Blockchain-oriented Software Engineering
CHASE	Cooperative and Human Aspects of Software Engineering
CI	Continuous Integration
CMMI	Capability Maturity Model Integration
CoEST	Center of Excellence for Software Traceability
DAO	Decentralized Autonomous Organization
Dapps	Decentralized Applications
DNS	Domain Name System
FAA	Federal Aviation Administration
GSE	Global Software Engineering
GT	Grounded Theory
ICSE	International Conference on Software Engineering
IEC	International Electrotechnical Commission
IoT	Internet of Things
IoV	Internet of Value
IPFS	InterPlanetary File System
IR	Information Retrieval
IS	Information Systems
ISO	International Organization for Standardization
LSI	Latent Semantic Indexing
MBT	Model-based Testing
MSP	Membership Service Provider
NaPiRE	Naming the Pain in Requirements Engineering
NDA	Non-disclosure Agreement
NDL	Neural Distributed Ledger
NFR	Non-functional Requirements
NSD	Norwegian Center for Research Data
OEM	Original Equipment Manufacturer
P2P	Peer-to-Peer
PICO	Population, Intervention, Comparison, Outcome
PM	Probability Model
PO	Product Owner
PoS	Proof-of-State

Acronyms

PoW	Proof-of-Work
RE	Requirements Engineering
RT	Requirements Traceability
RTM	Requirements Traceability Matrix
SC	Smart Contract
SDLC	Software Development Life Cycle
SE	Software Engineering
SHA	Secure Hashing Algorithm
SLR	Systematic Literature Review
SPI	Software Process Improvement
STXO	Spent Transaction Output
SWEBOK	Software Engineering Body of Knowledge
TSS	Text Semantic Similarity
TIM	Traceability Information Model
ULID	Universally Unique Lexicographically Sortable Identifier
UML	Unified Modeling Language
UTXO	Unspent Transaction Output
VSM	Vector Space Model

Contents

Preface	iii
Acknowledgment	v
Abstract	vii
List of Papers	ix
Acronyms	xi
Contents	xiii
List of Figures	xvii
List of Tables	xix
1 Introduction	3
1.1 Motivation	3
1.2 Research Goal, Objectives and Questions	6
1.3 Methodology	9
1.4 Contributions	10
1.4.1 C1: A list of RT challenges and their categories	10
1.4.2 C2: Blockchain-enabled SE use cases identification and categorization	10
1.4.3 C3: Process to implement blockchain in organizational settings	10
1.4.4 C4: Blockchain-enabled framework for trustworthy RT	11
1.4.5 C5: Development and validation of BC4RT prototype based on neural distributed ledger	11
1.5 Thesis Outline	12
2 Background	15
2.1 Blockchain Technology	15
2.1.1 Blockchain Data Structure	15
2.1.2 Blockchain Properties	18
2.1.3 Blockchain Generations	21
2.1.4 Blockchain Platforms	23
2.2 Software Engineering	32
2.2.1 Global Software Engineering	32

2.2.2	Requirements Engineering	33
2.2.3	Requirements Traceability: Definition and Technologies	34
2.2.4	Requirements Traceability Challenges	37
2.3	Blockchain and Software Engineering	39
2.3.1	Software Engineering for Blockchain Technology	39
2.3.2	Blockchain Technology for Software Engineering	40
3	Research Methodology	45
3.1	Design Science Methodology	45
3.2	Research Design: A Multi-Method Approach	46
3.3	Data Collection	47
3.3.1	Systematic Literature Review	47
3.3.2	Systematic Mapping	48
3.3.3	Interviews	50
3.3.4	Selection of Participants	51
3.4	Data Analysis	53
3.4.1	Grounded Theory	53
3.4.2	Content Analysis	58
3.5	Reflections on Methodological Quality	60
3.5.1	Validity	62
3.5.2	Generalizability	63
3.5.3	Ethical Considerations	64
4	Research Findings	65
4.1	Paper I: Blockchain-oriented Requirements Engineering: A Framework	65
4.2	Paper II: What have we learnt from the challenges of (semi-) structured requirements traceability?	66
4.3	Paper III: Software Engineering Applications Enabled by Blockchain Technology: A Systematic Mapping Study	68
4.4	Paper IV: A Blockchain-enabled Framework for Requirements Traceability	71
4.5	Paper V: Blockchain for Requirements Traceability: A Qualitative Approach	74
4.6	Paper VI: A Neural Blockchain for Requirements Traceability: BC4RT Prototype	77
4.7	Paper VII: Trustworthy and Collaborative Requirements Traceability: Validation of a Blockchain-enabled Framework	79
5	Discussion	83
5.1	Synthesis of Findings	83
5.1.1	RQ1: What are the Challenges of Implementing Requirements Traceability?	83
5.1.2	RQ2: How can Blockchain Technology be used in the Software Engineering Domain?	84

5.1.3	RQ3: Is it possible to build a blockchain-enabled framework for decentralized and trustworthy requirements traceability that can be used by organizations in interorganizational software projects?	86
5.2	Contributions	89
5.2.1	Contributions to Theory	89
5.2.2	Contributions to Practice	90
5.3	Limitations	90
5.4	Future Work	91
6	Concluding Remarks	95
	Bibliography	97
	Research Papers	116

List of Figures

- 2.1 Blockchain data structure, adapted from Belotti et al. [1] 17
- 2.2 Merkle tree procedure: Case of odd number of transactions,
Adapted from Belotti et al. [1] 17
- 2.3 Digital signature process (red key represents the private key, green
key represents the public key) 20
- 2.4 Order-Execute Architecture 25
- 2.5 Execute-Order-Validate Architecture 26

- 3.1 Research methodology, adapted from Hevner et al. [2] 46
- 3.2 Primary studies selection process 49
- 3.3 Systematic mapping methodology, adapted from [3] 49
- 3.4 Grounded Theory phases, adapted from Hoda et al. [4] 57
- 3.5 Representation of *key activities* category by means of Glaser’s
temporal/process family 58
- 3.6 Mind map to depict the emergence of the category "Challenges"
from underlying concepts and codes 59
- 3.7 Content analysis process, adapted from Elo and Kyngäs [5] 62

- 4.1 Systematic map visualization using bubble plot 71
- 4.2 Initial blockchain-oriented RT framework, published in [6] 72
- 4.3 Emergence of core category, related categories and concepts by
means of GT techniques 75
- 4.4 Framework to implement blockchain for RT in organizational
settings 76
- 4.5 Underlying blocks’ structure, published in [7] 79

List of Tables

- 1.1 Relations between papers, research phases, objectives, questions, and contributions 9
- 3.1 Demographics of blockchain experts and SE experts 52
- 3.2 Sample of the coding analysis process 61

Part I: Research Overview

Chapter 1

Introduction

“Ever tried. Ever failed. No matter. Try again. Fail again. Fail better.”

(Samuel Beckett)

Human beings have, for thousands of years, developed the ability to trace and have used it in various disciplines. Tracing was first used in animal hunting, later escalating to other disciplines ranging from epidemiology to metrology [8]. A relatively new discipline that employs traceability is software engineering (SE). Software engineering researchers have paid attention to requirements traceability (RT) for years, acknowledging its importance and benefits. Traceability was first identified as a topic of interest in SE in 1976 [9]. By the 1980s, commercial tools with support for traceability were introduced [10], followed by an extensive amount of published research in the late 90s. This research focused particularly on the RT problem [11] and factors hindering its use in practice [12]. In 2011, traceability researchers and practitioners formed the Center of Excellence for Software Traceability (CoEST) and identified in one of their technical reports eight challenges of traceability: purposed, cost-effective, configurable, trusted, scalable, portable, valued, ubiquitous [13]. The last challenge represents the grand challenge of traceability, as it requires addressing all the other challenges. The vision for traceability in 2035 embraces the concept of ubiquitous traceability, i.e., traceability that is everywhere, accessed by everyone without asking for it, since it will be incorporated into the SE process.

However, achieving ubiquitous traceability is not trivial, particularly in large-scale and complex software projects that involve distributed and cross-organizational teams, a variety of tools and a high number of ever-changing software artifacts generated by these tools. This thesis aims to contribute to the traceability body of knowledge and vision by exploiting the inherent features of novel technologies, such as blockchain technology to ensure the availability, integrity and trustworthiness of traceability information that is embedded into the distributed, and often global software engineering process. This chapter motivates the topic of this thesis, formulates the main research goal, objectives, and questions, introduces the research methodology adopted in the thesis, and outlines the main contributions of the thesis in relation to research phases, objectives, questions, and published papers. Finally, the chapter guides the interested readers by presenting the overall structure of the thesis.

1.1 Motivation

Today’s software systems are not only complex, but they also encompass an ever-increasing number of safety-critical functions [14]. The inappropriate

1. Introduction

specification, design, development and testing of such complex systems may lead to serious consequences, from major financial losses to life-threatening accidents [14, 15]. The complexity of such systems is also related to their size, e.g., up to approximately 100 million code lines, and 2000 pages of requirements have been reported in the automotive domain [14]. Another factor that contributes to increased complexity lies in the large number of evolving artifacts that are created by distributed stakeholders within the organization or across organizational boundaries, e.g., relations between original equipment manufacturer (OEM) and suppliers [14]. In this context, keeping track of all artifacts, their relations and versions is not a trivial activity, as it requires the establishment of standardized methods for traceability.

A variety of regulatory agencies in different domains have acknowledged the importance of traceability and have incorporated it into different software process improvement (SPI) models and standards, such as Federal Aviation Administration (FAA) DO-178C, ISO 26262, Automotive Software Process Improvement Capability dEtermination (ASPICE), and ANSI/AAMI/IEC 62304 [14, 15, 16]. FAA DO-178C mandates developers to provide evidence on the traceability of designs against requirements [15]. ISO 26262 is an automotive safety standard that requires traceability between safety artifacts, artifacts' versioning and unique identifiers [14, 15]. In addition, traceability requirements in the automotive domain are imposed by ASPICE which mandates traceability of all artifacts, not limited to safety-related artifacts [14]. Traceability requirements are also imposed in medical devices software development processes by different standards, such as ANSI/AAMI/IEC 62304 [15, 16]. Compliance to regulations has been considered as one of the main motivators to implement traceability in regulated domains [17].

In addition to compliance, the implementation of traceability brings benefits to organizations, as it facilitates the development of high quality software [17]. Regan et al. [17] explored a set of case studies to understand the motivations of organizations for implementing traceability. The findings revealed that traceability enables artifacts' reuse which allows software engineers to save time, effort and be more productive, as they do not need to reproduce all the artifacts. In addition, the findings suggested that while forward traceability facilitates the validation of requirements, backward traceability mitigates the so-called "gold plating" phenomenon in SE, i.e., excess functionality. Additional benefits of traceability, such as supporting project management, facilitating cost estimation of changes, and enhancing the understanding of the overall system have been constantly acknowledged by the traceability community [10, 18, 19, 20].

Despite the theoretical benefits, the adoption rate of RT has been surprisingly low [21, 22, 23, 24, 25, 26]. Previous studies have reported on the fact that traceability is often not implemented or implemented in an ad-hoc manner [14, 17, 24, 27]. Traceability is often considered an extra task to be carried out by developers who do not always perceive its benefits. The high perceived overhead/benefit ratio can be explained with the provider-user gap, as referred to in the second paper of this PhD thesis (P2). This gap means that the creators of trace links, e.g., developers create traces between implementation tasks and

source code commits, are not the ones who use the trace links, e.g., project managers use trace links to track the progress of the project. Hence, developers become demotivated to devote effort and time into creating and maintaining quality trace links and set a low priority to this task, resulting in incorrect and missing trace links. A few approaches have been proposed in literature to increase the motivation of developers to participate in traceability tasks. For instance, Maro et al. [14] proposed including gamification features into traceability tools, while Wohlrab et al. [28] explored collaborative traceability management and recommended the inclusion of voting features to allow stakeholders to agree upon related artifacts and indicate incorrect relations as a joint effort. Managing traceability as a collaborative effort can improve the perceived overhead/benefit ratio and encourage developers to contribute to traceability creation and maintenance. It is worthy to outline that trace links need to be maintained and updated once related artifacts change to prevent traceability deterioration. Artifacts' changes should be propagated to all affected stakeholders to enable them to update corresponding trace links.

Today's software is developed as a product of complex supply chains that rely on the collaboration of diverse, distributed entities throughout all the phases of the software development lifecycle, from requirements elicitation, to design, development, and maintenance. Tracing ever-evolving artifacts is difficult in co-located development, and unsurprisingly is even more challenging in multi-site environments. Cross-organizational and distributed entities need to share artifacts and have visibility over the software development lifecycle when collaborating for the development of complex software. To enable artifacts' sharing, Maro et al. [14] proposed the use of a centralized repository where all artifacts can be stored and accessed by legitimate stakeholders. However, centralized approaches cannot ensure that all stakeholders are working on the same reality, as artifacts can be tampered with. In addition, previous literature, as outlined by Yau and Patel [29], argued that in such approaches, it is not always possible to obtain the auditable, trusted and reliable history of software components, and consequently their corresponding trace links. Moreover, the artifacts created by distributed entities cannot always be trusted, as these entities may be related to competitors or groups with malicious intentions [29]. Other concerns about centralized solutions in distributed software development have been raised by Yau and Patel [29], for instance, the single point of failure, centralized authority, data tampering, restricted data ownership and lack of verification of added data.

The participation of third-party entities exacerbates these concerns, due to differences in data and artifacts sharing policies [29]. The complexity of enabling traceability increases in interorganizational software projects, due to differences in tooling and processes that are used for traceability management, conflicting organizational objectives, and organizational boundaries [30]. Conflicting objectives concerning traceability may lead to the creation of incompatible (e.g., with regards to type and granularity), thus unusable trace links. Finally, organizational boundaries restrict access to artifacts that contain sensitive data, impeding full visibility and consequently impeding the creation of complete and

trustworthy traceability. As advocated by Rempel et al. [30], it is necessary to ensure that traceability information is available and trustworthy across organizational boundaries. One way to approach these concerns is by considering a technology with inherent properties of decentralization, availability, and trust, such as blockchain. This context is the catalyst for the research described in this thesis, which explores the use of blockchain technology for decentralized, yet trustworthy RT in interorganizational software projects.

1.2 Research Goal, Objectives and Questions

The main goal of this thesis is to explore the use of blockchain technology for decentralized and trustworthy RT in interorganizational software projects. The underlying motivations for this goal lie in the inherent properties of blockchain technology: immutability, non-repudiation, decentralization, integrity and auditability (See Section 2.1.2), and the limitations of current centralized approaches for RT in interorganizational software projects (See Section 1.1). In order to achieve this goal, a set of 6 operational research objectives (RO) and their related research questions (RQ) were formulated, as follows:

- (i) **RO1.** Identify and classify challenges of implementing RT. This objective will be accomplished by reviewing and analysing relevant and recent scientific literature focused on RT.
- (ii) **RO2.** Identify and classify blockchain use cases in the SE domain and explore the benefits that the application of blockchain technology can bring to the SE landscape. This objective will be accomplished by reviewing relevant literature and mapping the two fields of blockchain technology and SE.
- (iii) **RO3.** Design a blockchain-enabled framework for decentralized and trustworthy RT that can be leveraged by organizations in interorganizational software projects. This objective will be accomplished by designing a framework with components that rely on relevant literature, that can be tailored to different organizational needs.
- (iv) **RO4.** Refine the framework by incorporating blockchain experts' feedback. This objective will be accomplished by conducting semi-structured interviews with blockchain experts from different domains and using their feedback to refine the framework and enhance its practicality.
- (v) **RO5.** Implement a prototype for decentralized, trusted, and scalable RT. This objective will be achieved by developing a prototype that enables the decentralized and secure storage of software artifacts, their related artifacts and changes, while retaining performance efficiency and scalability.
- (vi) **RO6.** Evaluate the proposed framework and prototype by means of SE experts' judgement. This objective will be achieved by conducting semi-structured interviews with SE experts with a broad academic and/or

professional expertise. These experts will be provided with the proposed framework and prototype prior to the interview process, and then they will answer questions regarding the usefulness, validity, and feasibility of the proposal, along with recommendations for further improvements.

These operational objectives aim to answer the following three research questions:

- (i) **RQ1.** What are the challenges of implementing requirements traceability? This research question addresses RO1 and focuses on the identification and categorization of RT challenges. While traceability researchers have devoted their efforts to identifying and addressing such challenges, to date many of the challenges remain unsolved. Our preliminary literature search carried out at the beginning of this PhD project found out that the last study that identified a comprehensive list of traceability challenges in requirements engineering was published in 2010 [31]. Although the authors elaborated on a set of natural, technical, economical, and social factors that challenge the implementation of RT practices, it is relevant to identify domain-agnostic RT challenges according to recent empirical studies in the field. Addressing RQ1 provided the underlying foundation of the discussion around the application of blockchain technology for trustworthy RT.
- (ii) **RQ2.** How can blockchain technology be used in the software engineering domain? This research question addresses RO2 and focuses on the identification of blockchain use cases in the SE domain and the benefits that this technology can bring to address SE issues. The importance of RQ2 lies in outlining the novelty of the blockchain-enabled RT topic, and in generating ideas inspired by previous approaches on how to address the topic. The preliminary search on blockchain-enabled SE suggested an increasing number of studies focused on addressing SE issues in blockchain-based software, e.g., Vacca et al. [32] (For additional studies, see Section 2.3.1), but a limited number of studies that use blockchain to address issues in the software industry. While accessing a comprehensive overview of previous blockchain-based approaches in SE was important within the scope of this thesis, our search revealed only one related study carried out in 2019 by Tariq and Colomo-Palacios [33]. Although this study provided us with a better understanding of the potential applications and benefits that blockchain can bring to the SE landscape, it is limited to research conducted up to 2018. Given that blockchain research is rapidly evolving and new approaches are emerging even in the SE field, a more recent overview on the topic was considered of high value for advancing our research work.
- (iii) **RQ3.** Is it possible to build a blockchain-enabled framework for decentralized and trustworthy requirements traceability that can be used by organizations in interorganizational software projects? This research question addresses the remaining objectives [RO3-RO6] and is answered by designing, refining, implementing, and evaluating a blockchain-enabled

framework for decentralized and trustworthy RT. To ensure trust among stakeholders in interorganizational software projects, it is important to enable them to work on the same reality and have a holistic view on software artifacts, their relations, changes, and the entire software lifecycle. This can be achieved by providing a reliable traceability knowledge base that does not need to be maintained by a central authority. In this regard, blockchain technology can be considered a suitable technology, due to its inherent properties of decentralization, immutability and integrity, among other properties (See Section 2.1.2). However, despite these promising advantages, conventional blockchain platforms face performance efficiency and scalability limitations [34, 35]. Therefore, this PhD thesis ought to investigate other platforms that offer decentralization and security, while being performance-efficient and scalable, should such platforms exist.

The research questions were answered in 7 papers ($P_{i,i=1-7}$) published during the three-year timeframe of this PhD project, as shown in Table 1.1. During the first year, three research papers were published (P1, P2, P3). The first research paper provided independent and constructive feedback on the PhD research project and future research directions. P1 was presented to the RE community and provided also an opportunity to interact with established researchers and practitioners in the software engineering community. This paper was part of the Doctoral Symposium session of the International Requirements Engineering conference 2020. Furthermore, P2 and P3 addressed RQ1 and RQ2 respectively, and were published in Scopus-indexed journals. The first year was important to acquire necessary knowledge about the topic of interest and plan on how to use the knowledge in the next steps. The last two years of the project focused on the core research question, i.e., RQ3 which entails different aspects: design, refinement, implementation, and evaluation of the proposal.

The initial blockchain-enabled framework for trustworthy RT was proposed in P4 which was published and presented in EuroSPI 2021 conference. This framework was refined by incorporating blockchain experts' feedback in P5 which was published in *Journal of Software: Evolution and Process*. As a proof-of-concept, a prototype that relied on neural distributed ledger platforms was developed and validated by using artifacts of an electronic health records application, named iTrust in P6 which was published and presented in EuroSPI 2022 conference. Finally, P7 evaluated the proposed framework and prototype by experts with a broad academic and/or professional experience in SE. The paper was submitted to the *Journal of Software: Evolution and Process*. The logical relations between research objectives, questions and published papers are presented in Table 1.1:

Table 1.1: Relations between papers, research phases, objectives, questions, and contributions

Papers	Research Objectives						Questions			Phase	Contributions					
	RO1	RO2	RO3	RO4	RO5	RO6	RQ1	RQ2	RQ3		C1	C2	C3	C4	C5	
P1*										Knowledge base						
P2	X										X					
P3		X						X				X				
P4			X						X	Build- Evaluate					X	
P5				X					X				X	X		
P6					X				X							X
P7						X			X							X

P1* entailed the publication and presentation of the PhD project plan at a doctoral symposium, but it did not tackle a specific research question

1.3 Methodology

This section provides a short introduction of the research methodology adopted throughout the PhD project. This thesis followed the design science methodology proposed by Hevner et al. [2], as it relies on building and evaluating artifacts that aim to achieve utility. This is in line with our core research question, i.e., RQ3 that aims to build blockchain-based artifacts for requirements traceability and evaluate their utility for interested organizations. The methodology consisted of two core phases that adopted different data collection methods and analysis techniques. In what follows, these phases are explained briefly:

- (i) *knowledge base*. In this phase, solid research foundations were built within the areas of global software engineering, requirements traceability and blockchain technology. The first three papers (P1, P2, P3) were written within the scope of this phase (See Table 1.1). The first paper was a description of the PhD project, published and presented during the doctoral symposium session of the main conference in requirements engineering. Furthermore, P2 adopted a systematic literature review (SLR) approach based on Kitchenham [36]’s guidelines for systematic literature review studies in software engineering, while P3 adopted a systematic mapping approach based on Petersen [3]’s guidelines for systematic mapping studies in software engineering.
- (ii) *build-evaluate*. The knowledge acquired during the first phase was used to build the initial blockchain-enabled framework for requirements traceability (P4). In order to refine the framework with a more practical perspective, semi-structured interviews were carried out with blockchain experts, and the data was analysed using grounded theory analysis techniques (P5). The refined framework led to the development of blockchain for requirements traceability (BC4RT) prototype (P6) which in turn was evaluated by SE experts (P7). The data was collected by carrying out semi-structured interviews with SE experts and analysed using content analysis technique (P7).

Data collection methods and analysis techniques are explained in detail in Chapter 3.

1.4 Contributions

1.4.1 C1: A list of RT challenges and their categories

The first contribution addressed the first research question (RQ1) by means of P2. A list of 21 challenges of implementing RT in practice was created by carrying out a SLR on 70 related papers published during the period 2009-2019. To facilitate the elaboration of these challenges, they were classified into 5 main categories: technological, human factors, organizational, communication and collaboration, and regulatory challenges. While the analysis of the data extracted from the studies had purely qualitative nature, the number of studies that addressed each of the challenges was calculated to pinpoint existing research gaps that in turn, suggested future research directions. The main findings are presented in Chapter 4, and future research directions are recommended in Chapter 5.

1.4.2 C2: Blockchain-enabled SE use cases identification and categorization

The second contribution addressed the second research question (RQ2) by means of P3. A systematic mapping study was conducted to provide a holistic overview of blockchain use cases in SE. The analysis of existing relevant literature indicated a set of blockchain-enabled use cases grouped into 8 SE knowledge areas: software requirements, SE process, software testing, software quality, software maintenance, software configuration management, SE management and professional practice. The identified applications were also analysed from a technical perspective in terms of blockchain platforms and consensus mechanisms used. These use cases are explained in Chapter 4 and P3. Further, the use cases were mapped with research types and contributions to reveal existing gaps and suggest future research directions. Finally, the thesis contributes by mapping blockchain properties and SE challenges addressed to emphasize the benefits that blockchain technology can bring to the SE field, for instance, decentralization, transparency and trust, immutability and data security, anonymity, non-repudiation and automation by means of smart contracts/chain code. This contribution is particularly important, as it can encourage further research efforts in the emerging blockchain-oriented SE field.

1.4.3 C3: Process to implement blockchain in organizational settings

The third contribution addressed the third research question (RQ3) by means of P5. Semi-structured interviews with a variety of blockchain experts were carried out and grounded theory techniques were applied to the transcripts' data. The iterative analysis of data unveiled the process for implementing

blockchain technology in organizational settings. This process consisted of the following key activities that should be performed in an iterative manner: identify business needs, perform feasibility analysis, blockchain platform selection, test through prototypes, shift from prototyping to production and scaling. In addition to key activities, a set of success factors and challenges were identified and elaborated in detail in P5. Finally, the findings of this study contributed by revealing reasons behind the organizational resistance to change with regards to implementing blockchain technology, such as innovation-production gap, centralized mentality, and conservative management. In this thesis, the findings related to the implementation process of blockchain in organizational settings were used to refine the proposed blockchain-enabled framework for RT. Moreover, the findings can be also useful to guide practitioners who are eager to leverage the potential of blockchain technology.

1.4.4 C4: Blockchain-enabled framework for trustworthy RT

The fourth contribution addressed the third research question (RQ3) by means of P4 and P5. The initial blockchain-enabled RT framework was proposed in P4. To date, this is the first approach that uses blockchain to keep track of software artifacts and trace links in interorganizational software projects. The framework contributes to the RT field by providing stakeholders with a holistic and trusted view of software artifacts and their trace links, offering an incentive mechanism for stakeholders to create quality trace links, enabling collaborative traceability management by proposing the use of voting mechanisms to jointly agree upon the authenticity, accuracy and quality of trace links, facilitating the understanding of traceability information through query services for primitive trace links and composite traceability paths, and enhancing the understanding of traceability information by means of interactive graphical representation of trace links. Given that the components of the initial framework emerged from literature relevant to the fields of RT and blockchain technology, blockchain experts were interviewed in P5. The goal of this approach was to incorporate blockchain experts' feedback into the framework to refine it in a more practical manner.

1.4.5 C5: Development and validation of BC4RT prototype based on neural distributed ledger

The last contribution addressed the third research question (RQ3) by means of P6 and P7. A neural blockchain prototype for reliable and collaborative traceability management was proposed in P6. This is one of the first prototypes that relies on the concept of neural distributed ledger (NDL) by using the NDL ArcaNet platform (See Section 2.1.4). Differently from conventional blockchain platforms, neural distributed ledgers enable the storage of software artifacts of any type or size, and their changes in a decentralized, scalable, and efficient fashion, while retaining security. The NDL ArcaNet-enabled prototype creates tokens for each project and each requirement within the project, keeps track of

tokens' lifecycle and enables the certification of operations on tokens without using inefficient consensus mechanisms (See Chapter 4 and P6). The prototype enables stakeholders of the software development lifecycle with a trustworthy view on what/how/when and by whom software artifacts are created and/or updated. This can likely lead to enhanced communication, coordination, and trust among software development lifecycle entities. In turn, the improved trust can have a positive impact on software development efficiency and software quality. Finally, the prototype was presented to SE experts who stated their perceptions and recommendations on how to enhance the prototype's practicality and validity. These recommendations pinpointed promising future research avenues which are discussed in Chapter 5.

Table 1.1 depicts the relations between research phases (See Section 1.3), research objectives and questions (See Section 1.2), papers (See List of Papers), and contributions (See Section 1.4).

1.5 Thesis Outline

The thesis is organized into 6 chapters that are described in the following section: **Chapter 1** introduces and motivates the topic of the thesis by outlining existing challenges and the need to find innovative approaches to address such challenges. In addition, the chapter presents the main research goal, objectives, questions, and the methodology adopted to address the research questions and objectives. Finally, the chapter outlines the contributions of the thesis and relates these contributions to their respective research phases, objectives, questions, and published papers.

Chapter 2 frames the background around key concepts used in this thesis. First, the chapter describes blockchain data structure and provides an overview of the core blockchain properties, generations, and platforms. Second, the chapter defines the software engineering field, and discusses global software engineering, requirements engineering and requirements traceability. Particular attention is paid to defining requirements traceability, describing RT technologies, and identifying RT challenges. This is in line with the research problems formulated in Section 1.2. Finally, the chapter unveils the bidirectional relationship between blockchain technology and software engineering.

Chapter 3 explains thoroughly the design science methodology adopted in this thesis. A set of data collection and analysis techniques, such as systematic literature review, systematic mapping, semi-structured interviews, grounded theory, and content analysis are described based on scientific literature and the specific implementation in this thesis. The chapter ends with reflections on ethical issues and measures undertaken to minimize such issues.

Chapter 4 presents an overview of the 7 research papers published within the scope of this PhD thesis. This overview consists of the purpose of the paper, research approach, main findings, and contributions.

Chapter 5 synthesizes the main findings in relation to the research questions formulated in Section 1.2. In addition, the chapter discusses limitations of the

work and recommends promising future research dimensions that deserve further research efforts.

Chapter 6 summarizes the work carried out in this thesis and outlines future research work.

Interested readers are recommended to read the full texts of the 7 research papers which are included in **Part II**.

Chapter 2

Background

2.1 Blockchain Technology

“The first generation of the digital revolution brought us the Internet of Information. The second generation powered by blockchain technology is bringing us the Internet of Value: a new, distributed platform that can help us reshape the world of business and transform the old order of human affairs for the better”.

(Don Tapscott and Alex Tapscott [37])

Blockchain technology can be defined as a distributed ledger technology that enables transactions to be read, validated, and stored in a chain of blocks [1], or as a meta-technology, meaning technology that is created as a result of the cohesion among several technologies [38]. Although blockchain emerged with the seminal paper on Bitcoin in 2008 [39], the majority of the underlying technologies have been proposed many years earlier [40]. While the record keeping idea goes back to ancient Mesopotamia, the concept of distributed databases goes back to the 1970s [1, 40]. Furthermore, the idea of linking blocks in an immutable manner by means of a cryptographic hash function was proposed in 1979 by Ralph Merkle [41] who explained the concept of linking information in a hash tree structure, later coined as “Merkle hash tree”, in his dissertation at Stanford. Finally, despite the fact that the domain of trading has, for a long time, adopted rules for the execution of contracts’ terms, it was only in 1994 that Szabo coined the term “smart contracts” [42]. Smart contracts became popular with the emergence of the Ethereum platform in 2013 [43]. In what follows, an overview of the main blockchain concepts is provided and organized into four sections: blockchain data structure, blockchain properties, blockchain generations, and blockchain platforms.

2.1.1 Blockchain Data Structure

Blockchain ledger represents the history of validated transactions that are organized into blocks; each of the blocks is linked to the previously validated block by incorporating the hash value of the previous block header. This linked structure enables one to trace back to the first (parentless) block of the ledger which is referred to as the genesis block. The structure of the blocks depends on the blockchain platforms, however the conventional block structure is depicted in Figure 2.1, and consists of the following elements [1]:

- (i) **outer header** specifies the blockchain platform and block size. The outer header consists of an identifier for the specific blockchain platform, which

2. Background

is named “magic number”, and the block size field that indicates the maximum number of block bytes.

(ii) **block header** consists of fields that provide information on the previously validated block and the validation process. The main fields are:

- **Block version** indicates which validation rules to follow and it is useful to keep track of updates throughout the specific blockchain protocol.
- **Previous hash** is the value generated as a result of applying the hash function to the previous block header which serves as an input. The importance of this field lies in enabling a linked immutable structure of blocks.
- **Timestamp** serves as a proof that the block was mined and validated at a specific instance of time, therefore this field ensures the authenticity of any block.
- **Target** indicates the computational power required for the mining process; the higher the target, the higher the mining complexity.
- **Nonce** is a fixed-length number that should be found by miners for the validation of the block.
- **Merkle tree root** is the hash value that is generated as a result of applying a hash tree procedure on the set of transactions.

Figure 2.2 shows that transactions are hashed in pairs in an iterative manner. In the simple case scenario presented in Figure 2.2, the hash function is first, applied on the contents of transactions. Second, the hash function is applied on pairs of transactions. In this case, there is an odd number of transactions, therefore the last pair is created between hash5 and its duplicated value. The process continues iteratively until the Merkle tree root is generated. Duplicated hash values are marked in blue in Figure 2.2. The Merkle tree root is crucial in ensuring the integrity of the transactions that are included in the block.

(iii) **block body** contains a list of all transactions that take part in the Merkle tree procedure to generate the Merkle tree root which is part of the block header. In addition, the block body consists of the transaction counter field that counts the number of transactions in the block. It is noteworthy that the block body is directly dependent on the outer header because the block size influences the number of transactions that can be contained in the block body.

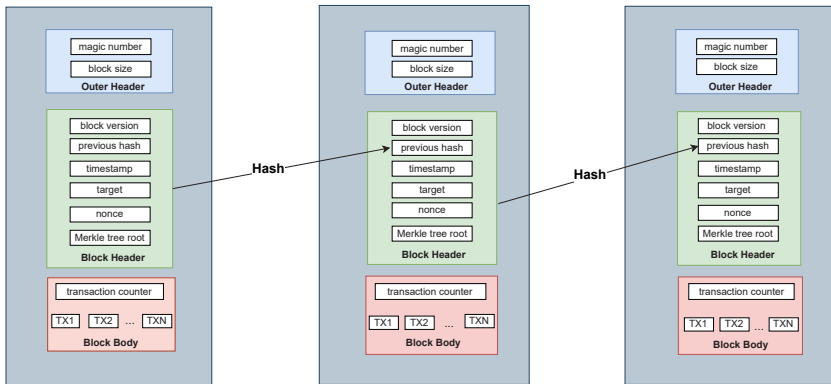


Figure 2.1: Blockchain data structure, adapted from Belotti et al. [1]

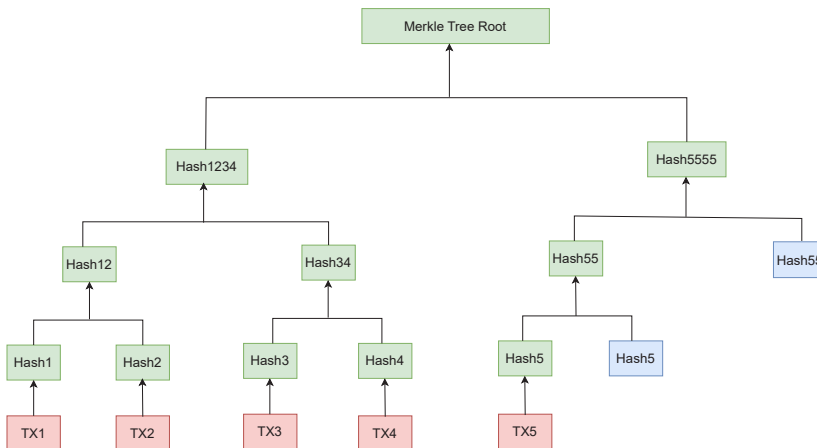


Figure 2.2: Merkle tree procedure: Case of odd number of transactions, Adapted from Belotti et al. [1]

2.1.2 Blockchain Properties

The combination of different technologies that were explained in Section 2.1.1 enables a set of dependent blockchain properties, as follows [1]:

- (i) **Decentralization.** The main utility of blockchain lies in eliminating the need to entrust centralized authorities for maintaining the ledger. Instead of that, data is stored and shared in a peer-to-peer (P2P) manner. In fact, Bitcoin introduced the decentralization of controlling power from few to all participants in a monetary system, as an important security mechanism. Decentralization is considered a security mechanism, because it requires the malicious attacker to compromise over half of the consensus power to significantly harm the system [44]. Despite the importance of decentralization from a security perspective, recent studies, e.g., [45], have noticed increased centralization in public blockchains, such as Bitcoin and Ethereum, which in turn impacts the security of these platforms. As stated by Sai et al. [46], decentralization is not provided by design, instead it is a non-deterministic guarantee, as a result of the fusion of cryptography, distributed systems and incentive mechanisms.

Recently, Sai et al. [45] conducted a SLR with 89 papers published from 2009 to 2019 on the topic of centralization of public blockchain platforms. In addition, the authors performed expert interviews to refine the taxonomy of centralization. The analysis of their results indicated that the top 4 mining pools, i.e., consortiums of participants working in groups [47], aggregate 50.36% of controlling power in Bitcoin, and 63% of controlling power in Ethereum [45]. This mining-centralization raises security concerns, as the dishonest behavior of only a small number of entities, for instance heads of 4 main mining pools, can threaten blockchain platforms [45].

In what follows, additional types of centralization are briefly explained:

- *governance centralization.* There is a central control over governance of blockchain platforms that may concentrate the decision-making power to a few entities. In this regard, two main cases can be mentioned: Bitcoin's core developers made the decision to reduce the minimum transaction fee in an unilateral fashion [48], and Ethereum hard fork as a result of the Decentralized Autonomous Organization (DAO) attack [49].
- *wealth concentration* which can be used to increase the cost of successful transactions. An example to illustrate this point is the iFish attack on Ethereum platform in which attackers produced a high number of transactions with high fees in a short period of time. The impact of wealth concentration is that already rich nodes tend to increase their nodes faster than smaller nodes, referred to as the "rich get richer" paradigm [50].
- *centralized wallets.* The application of wallets has been considered a single point of failure and security threat [51], since their development

and maintenance is often performed by centralized organizations. A more detailed explanation of the centralization of public blockchain platforms can be found in [45].

Indeed, a higher level of centralization is present in permissioned blockchain platforms with the inclusion of trusted participants. However, permissioned blockchain platforms rely on decentralizing intermediaries, along with their roles, as opposed to cutting out third-parties completely which is the case of permissionless platforms [1]. A more detailed description of blockchain platforms is presented in Section 2.1.4.

- (ii) **Immutability.** This property is ensured by the structure of blockchain as a linked list constructed by blocks that contain the hash of the previous block. Sagar Bharadwaj et al. [52] explained the immutability of blockchain with a simplified scenario: Suppose an attacker attempts to tamper with the contents of block n . This implies the need to recompute the hash of block n . Due to the collision resistance property of hash functions, this recalculation of the hash of block n generates a new hash, that will be present in the following block $n+1$. Hence, the hash of block $n+1$, along with all the following blocks would need to be recalculated. In order to write history and create new valid blocks, the attacker would need the majority of mining power of the network in permissionless blockchains. In permissioned blockchains, 1/3 of the network is able to launch a successful attack and modify blockchain data. This indicates that the structure of blockchain complicates the successful execution of such attacks, yet it does not eliminate the risk with absolute certainty.

Interestingly, critical researchers [53] have considered immutability as a desired, yet non intrinsic property of blockchain. In their critical analysis, Conte de Leon et al. [53] shed light into the misconception that blockchain is unchangeable. One event that disproves such claims was Ethereum DAO hard fork which moved all tokens to a “withdrawal-only” contract stored on Ethereum blockchain [49]. This software was implemented by the majority of miners, while the others split from the main chain of blocks. Thus, in this case the distributed ledger was updated to erase the DAO. In the case of blockchain systems that rely on Proof-of-Work (PoW) consensus algorithms, significant computational work is required to modify the data, dependent on the strength of the hash function used. Hence, the term “Mutable-By-Hashing-Power” has been proposed in literature [53], as a refinement of the term “immutable”.

- (iii) **Integrity, authenticity and non-repudiation.** Blockchain guarantees integrity, authenticity, and non-repudiation by combining hashing, asymmetric cryptography, and digital signature schemes. Data hashing is a mathematical operation that takes as input data of arbitrary length and generates a fixed length output, in an irreversible manner. The cryptographic hash property of irreversibility or pre-image resistance ensures that

2. Background

the data is not tampered with during its transmission, i.e., data integrity. In addition, broadcasting the public key of the sender ascertains that the origin of the transaction is what it claims to be, i.e., authenticity, while the non-repudiation property is ensured by the signing procedure that proves the sending action. In what follows, the digital signature process is explained and depicted in Figure 2.3:

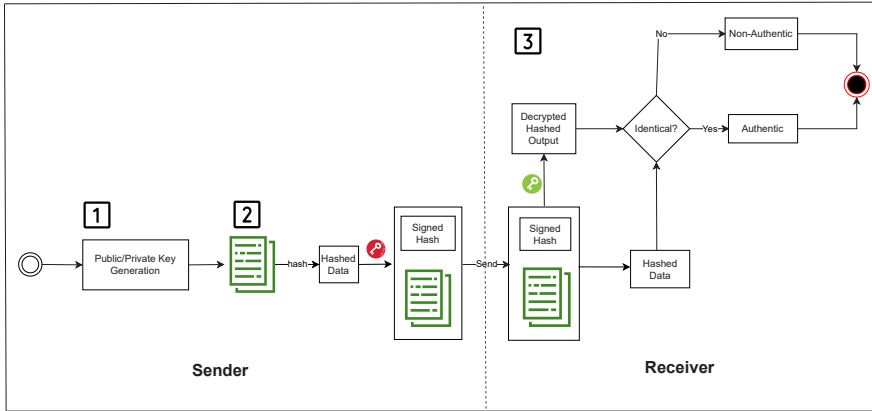


Figure 2.3: Digital signature process (red key represents the private key, green key represents the public key)

First, blockchain users generate a key pair that consists of a private key and a public key. While the former is used by the sender to sign transactions, the latter is used by the receiver to verify the integrity and authenticity of the data. Second, the sender hashes the original data and signs the hash output with the sender's private key. The role of hashing lies not only in ensuring the integrity of transmitted data, but also in generating compressed data of the same format which improves the efficiency of signatures. The signed hash along with the original, uncompressed, and unsigned data are sent to the receiver. Third, the receiver decrypts the signed hash by using the sender's public key and generates the decrypted hashed output. Moreover, the receiver hashes the original, unsigned data and produces the recomputed hash of the original data. Finally, the decrypted hashed output is compared with the recomputed hash of the original data. The identification of identical values verifies the authenticity of the sender and data integrity.

- (iv) **Automation.** An important feature of blockchain platforms is enabling smart contract deployment and execution to allow the automation of any business functionalities [1]. The conventional definition of a contract entails an agreement between parties to perform or not perform a specific action in exchange for something. These parties must trust one another to fulfill

the obligations specified in the contract. The innovative element that smart contracts bring is removing the need for trust between involved parties. Smart contracts are able to execute the terms of a contract in a distributed and trustless environment because they are both defined and enforced by the code in an automatic manner [54]. In a way, smart contracts solve common problems by minimizing the human involvement, and consequently enabling the trust in the system rather than in humans.

- (v) **Auditability.** Transactions stored on blockchain are visible and transparent to all blockchain participants. This enables traceability of blockchain operations via audits. The auditability property holds true not only in permissionless blockchains, but also in permissioned blockchain platforms such as Hyperledger Fabric. In such platforms, auditability is fulfilled at the channel level [1].

Given the intrinsic complexity of such a distributed system, proving the correct implementation of the aforementioned properties is a significantly hard endeavour, as acknowledged by the software engineering and distributed systems communities [53]. Despite the current challenges, researchers, e.g., [53] remain optimistic about future efforts devoted to mathematically prove that these desired properties hold under specific conditions, at a given point in time with measured certainty.

2.1.3 Blockchain Generations

Previous literature has identified four main generations of blockchain, based on the applications of this technology [54, 55]. These generations are explained in the following section.

- (i) **Blockchain 1.0.** The first generation of blockchain entails cryptocurrency applications, such as Bitcoin. The main utility of digital currencies lies in enabling transactions to be sourced and completed between individuals over the Internet removing the need for intermediaries [54]. In this way, payment fees are reduced, along with the waiting time for transfer, due to the fact that users are able to receive funds in their digital wallets in near real-time [55]. Cryptocurrencies other than Bitcoin are referred to as altcoins and they enable resources' allocation and trading globally in a completely decentralized manner, which goes beyond currency and payments. The potential of cryptocurrencies as programmable money has paved the way for the next generation of blockchain applications.
- (ii) **Blockchain 2.0.** Previous literature considers blockchain 2.0 as applications in other areas of finance, beyond currency and payments. In fact, the financial domain handles sensitive data, therefore enhanced security and data integrity must be ensured. Blockchain can be used to register and transfer different types of assets reliably in areas, such as supply chain finance, security trading, and anti-counterfeiting [55]. In his book

2. Background

on blockchain technology, Swan [54] explained in detail the main concepts that constitute the second generation of blockchain technology: smart property, smart contracts, decentralized applications (Dapps), and DAOs. The smart property concept entails property, in terms of physical assets (home, car, computer) or intangible assets (reservations, copyright), that is registered on the blockchain, and the control of its ownership is performed via smart contracts. Smart contracts are computer programs stored on blockchain that enable automatically solving problems by minimizing the need for trust between parties. Over time, the complexity and autonomy of smart contracts has increased. Smart contracts with their ability to perform programmed and self-programmed operations on a blockchain have become self-contained entities [54]. A set of smart contracts can encode management, operations and business rules of real-world physical organizations, thus enabling the model of autonomous organizations that run without human involvement. These new organizational forms are referred to as DAOs. Blockchain 2.0 projects that incorporate these features include, but are not limited to: Ethereum, Ripple, Mastercoin, and NXT [54].

- (iii) **Blockchain 3.0.** The evolution of blockchain technology enabled its application beyond the financial domain, in other domains, such as healthcare, government, and arts. One of the first non-currency applications of blockchain is Namecoin, as an alternative Domain Name System (DNS) that enables anyone to publish non-censored information on Internet [54]. Other blockchain 3.0 applications include digital identity verification (e.g., OneName, BitID), registering intellectual property and offering attestation services by means of hashing and timestamps (e.g., Proof-of-Existence), providing governmental services in a decentralized, cheaper and more efficient manner, for instance, voting, registration of will, marriage contracts, and land deeds [54]. An essential component of blockchain 3.0 is the incorporation of tokens. Tokens represent proofs of digital rights, such as personal identity, stocks, coupons, academic diplomas, and receipts [55]. In the blockchain context, tokens have been recognized, due to Ethereum's ERC20 standard [55]. While blockchain has been considered the back-end technology of the new era, tokens have been considered its front-end economic facet, and their combination has the potential to bring significant societal changes [55].
- (iv) **Blockchain 4.0.** Scientific literature failed to provide a unanimous definition of what constitutes the new generation of blockchain technology [56]. Angelis and Ribeiro da Silva [57] outlined the integration of artificial intelligence (AI) with blockchain technology as the main feature of the new wave of blockchain technologies. At first glance, AI and blockchain technology can be seen as two opposite sides of the technology spectrum. While AI expresses uncertainty by means of probabilistic theory, blockchain makes use of a deterministic hashing function to generate the same irreversible results when inputs do not change. However, these technologies

can be used jointly, for instance, blockchain can be used to feed data into AI systems and to store its outputs, due to blockchain's inherent ability to ensure data accuracy and trustworthiness [57]. Arenas and Fernandez [58] perceived the new generation of blockchain platforms as relying on the concept of blockchain-as-a-service (BaaS). This was also confirmed by the belief of professional literature [59] that fourth generation blockchains will focus on achieving higher efficiency, improved scalability, accessibility, and consequently mass adoption.

2.1.4 Blockchain Platforms

Since the advent of Bitcoin in 2008, a variety of blockchain platforms have emerged. Although these platforms have been developed targeting different application fields and use cases, they can be classified based on their network accessibility into two core groups: permissionless and permissioned platforms. The former allows anyone to access the network, read and write transactions, and participate in the consensus mechanism. The latter may restrict only rights on writing (validation), or rights on both writing (validation) and reading (access). Permissioned blockchain can be further categorized according to the nature of participants into: (i) private blockchains with intraorganizational participants, and (ii) consortium blockchains which entail a joint effort among several organizations with a common business goal or need. In what follows, the major blockchain platforms are described.

- (i) **Bitcoin.** Bitcoin is a permissionless and public blockchain platform, that was designed to serve as a public payment system [1]. While digital currencies have been around for decades, the novelty of Bitcoin lies in the combination of a simple consensus protocol that creates an ever-growing distributed ledger because nodes aggregate transactions into a block every 10 minutes, with PoW mechanism that allows nodes to gain participation rights in the system [43]. The basis of this mechanism lies in the fact that Bitcoin adopts a CPU-bound function, which is referred to as SHA-256 hash function. To prove that participants are real identities, they need to carry out work. This work entails a cryptographic puzzle, which increases computational costs artificially. Consequently, the PoW mechanism plays an important role in preventing Sybil attacks, as the verification ability depends upon the computational power, rather than the number of identities that can potentially be fake [60]. The underlying nature of miners indicates that they are profit-seekers, therefore they try to solve the cryptographic puzzle by trying different nonces until finding a solution. This mining process is costly in terms of mining hardware costs, and energy costs.

From a technical viewpoint, Bitcoin can be considered a “state transition system” which consists of state and a state transition function [43]. The function takes as input: (i) state (S) which entails bitcoins' ownership status or the collection of unspent transaction outputs (UTXO), and (ii)

2. Background

a transaction, and generates a new state or generates an error. The transaction consists of a hash value that identifies the transaction (txHash) and a list of inputs and outputs [60]. Inputs contain the hash of the previous transaction as identification of that transaction (prevTxHash), an index of the transaction's output, and the signature of the owner of the unspent transaction output. Outputs contain the amount of coins to be transferred and the destination's Bitcoin address. It is noteworthy that it is not permitted to reference the same output twice, an attempt that is referred to as double spending. Therefore, each output of a transaction in Bitcoin can be considered either UTXO if it has not been referenced by another transaction, or a spent transaction output (STXO). Furthermore, the state transition function can generate an error in three cases:

- when the referenced UTXO does not exist in S , which indicates an attempt to spend coins that do not exist,
- when there is a mismatch between the provided signature and the real owner of UTXO, which indicates an attempt to spend coins that are owned by other people,
- the sum of all inputs UTXO is less than the sum of all outputs UTXO. In other cases, the function returns a new state that adds all output UTXO and removes all input UTXO.

A more detailed overview on Bitcoin can be found in the technical survey of Tschorsch and Scheuermann [60].

$$F(S, Tx) \rightarrow S' \text{ or } ERROR$$

- (ii) **Ethereum.** Ethereum is a cryptocurrency-based, and permissionless blockchain platform, which was designed to enable anyone to write smart contracts and Dapps, by means of its built-in Turing-complete programming language, referred to as Solidity [43]. Ethereum uses a PoW-based consensus mechanism, named Ethash that differs from the PoW mechanism used by Bitcoin, due to its reliance on the concept of “memory hardness”. This concept means that the search for nonces requires significant memory and memory access bandwidth [1], hence it aims to prevent the simultaneous identification of nonces by using the memory in parallel [61]. Recent efforts have been devoted to migrating towards the more secure and efficient proof-of-stake (PoS) consensus mechanism with Casper implementation [62].

The state in Ethereum consists of objects, referred to as accounts that enable participants to create transactions and/or contracts, and mine the crypto-token namely Ether. These accounts contain the following fields: nonce or counter to ensure that transactions are processed only one time, current ether balance, contract code, and storage [43]. Moreover, there are two types of accounts in Ethereum [43]:

- a) *externally-owned accounts* that do not contain code and are controlled by private keys, and
- b) *contract accounts* that contain code that is activated once the account receives a message.

Furthermore, Ethereum adopts state machine replication, following an “order-execute” architecture (See Figure 2.4) which consists of first ordering transactions, and then broadcasting and executing them on all nodes in a sequential manner. A more detailed description of this blockchain platform can be found in the Ethereum’s whitepaper published in 2013 by Buterin [43].



Figure 2.4: Order-Execute Architecture

- (iii) **Hyperledger Fabric.** Hyperledger Fabric is a permissioned and private blockchain platform that runs distributed applications developed by means of general-purpose programming languages, such as Go, Java, Node.js. The execution of these applications occurs consistently across different nodes, thus it gives the impression that it occurs on a single blockchain computer that is distributed globally [63]. Furthermore, smart contracts in Fabric are referred to as chaincode and are written in Golang in the first version of Fabric, and in Javascript in Fabric v1.1 [1]. Developers use chaincodes to define and track the lifecycle of assets, to manage business contracts, and to create Dapps that are managed collectively [1]. It is noteworthy that in Fabric, isolation between chaincodes is ensured by channels which are independent instances with their specific set of rules and policies, and do not exchange data with other channels [1]. The concept of channels can be important when the network participants are competitors who do not want their transactions to be known to all the

2. Background

other participants [63]. Moreover, Hyperledger Fabric is underpinned by an extensible and modular architecture [63], which means that the platform supports pluggable implementations of consensus and membership services [1], thus it can be customized to a variety of use cases.

Due to its permissioned nature, members register via a Trusted Membership Service Provider (MSP) which maintains the identities of all types of nodes, and issues credentials for authentication and authorization [63]. Nodes in Fabric can have one of the following roles [1, 63]:

- Clients submit proposals for execution, contribute in orchestrating the execution stage, and broadcast transactions for the ordering stage.
- Peers maintain the ledger and the state. They can play two important roles: execute transactions and validate transactions. The execution of transactions is performed by a subset of peers named endorsers.
- Orderers do not participate in the stages of execution and validation, but they establish collectively the order of all transactions in Hyperledger Fabric. This separation of roles enables the modularity of consensus in Fabric and facilitates the replacement of consensus mechanisms [63].

Differently from Bitcoin and Ethereum, Hyperledger Fabric adopts an “execute-order-validate” architecture, as depicted in Figure 2.5.

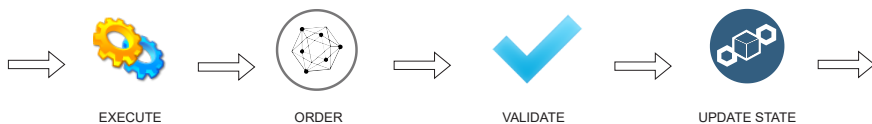


Figure 2.5: Execute-Order-Validate Architecture

The three phases of the transaction flow are explained, as follows [60]:

- a) *Execute*. In this phase, clients sign and submit transaction proposals to endorsers whose role is to execute transaction proposals. Transaction proposals contain the following set of elements: client's identity, transaction payload or operation that should be executed, parameters, chaincode identifier, nonce in the form of a counter or random value that can be used only once by a specific client, and a transaction identifier. Endorsers execute the transaction payload or the chaincode operation that has been installed on the distributed ledger. It is worthy to mention that the chaincode is executed in isolation from the main endorser process, in a Docker container.

Each endorser executes the transaction proposal, without being synchronized with the other peers, and yields writeset representing the state updates in terms of modified key-value pairs, and readset that represents the dependencies of the execution of the transaction proposal in terms of keys read during execution and their versions. The execution is followed by cryptographically signing the message (readset and writeset), referred to as endorsement. Endorsements are sent to the client who collects them until the endorsement policy is fulfilled. The endorsement policy requires all endorsers to generate identical outputs, i.e., readset and writeset. Once the client has accumulated the required endorsements, the client creates the transaction and sends it to the ordering service.

- b) *Order*. In this phase, the client sends the transactions consisting of transaction payload or chaincode operation, along with its parameters, transaction metadata, and endorsements to the ordering service. The role of the ordering service is to construct the order of submitted transactions for each channel. The ordering service broadcasts endorsements to all peers, achieving in this way consensus on transactions, even in the presence of faulty orderers. In addition, the ordering service groups transactions into blocks which is a technique adopted in fault-tolerant broadcasts to enhance the throughput of the broadcast protocol [63]. It is worthy to outline that the ordering service does not maintain blockchain states and does not execute or validate transactions. This implies an important feature of Fabric which is the separation of consensus from execution and validation.

The ordering service supports two operations that are called by clients: (i) broadcast (Tx) is invoked to disseminate transactions consisting of transaction payload and client signature (ii) $B([tx_1, tx_2, \dots, tx_n], i, h_{i-1}) \leftarrow deliver(i)$ is invoked to retrieve block B with the list of transactions $[tx_1, tx_2, \dots, tx_n]$, sequence number i, and the hash value of the block with sequence number i-1. The ordering service enables delivered blocks to be ordered by ensuring two main safety properties [63]:

- *agreement property*. For any two blocks B and B' with respective sequence numbers i and i', delivered such that $i == i'$, then

2. Background

- $B == B'$ holds true.
- *hash chain integrity property.* If a client has been delivered a block $B([tx_1, tx_2, \dots, tx_n], i, h)$, and another client has been delivered block $B'([ty_1, ty_2, \dots, ty_m], i + 1, h')$, then $h' == H(B)$ holds true.
- c) *Validate.* The dissemination of delivered blocks to peers can be done directly by the ordering service or by means of a scalable built-in gossip service. Afterwards, the blocks go through three main validation steps:
- i. All transactions of a specific block are assessed in parallel against the endorsement policy. Based on the results of this assessment, the transactions can be marked as valid or invalid.
 - ii. All transactions of a specific block are checked against in a sequential order for potential read-write conflicts. The keys' versions in the readset field of each transaction are compared with the existing locally stored ledger. The inconsistency of versions indicates the invalidity of the transactions.
 - iii. The block is appended to the ledger that is stored locally by writing key-value pairs in writeset fields, and the blockchain state is updated accordingly. It is interesting enough that in Fabric, the ledger contains the results of the first two validation steps, including invalid transactions. This occurs due to the design of Fabric, in which the chain of blocks is first produced by the ordering service, and the validation is carried out by peers afterwards. Although this feature is not available in other blockchain platforms, such as Bitcoin and Ethereum, it may be useful when audits of invalid transactions are needed. A detailed and comprehensive overview on Hyperledger Fabric, its architecture and design decisions rationale can be found in the paper published by Androulaki et al. [63].
- (iv) **NDL ArcaNet.** Blockchain technology has enabled the shift from the Internet of Information towards the Internet of Value (IoV) [64]. Vadgama et al. [64] defined IoV as “*the instant transfer of assets that can be expressed in monetary terms over the Internet between peers without the need for intermediaries*”. This instant transfer of assets that have a value can be achieved by means of blockchain, due to its inherent capabilities. Blockchain is able to create digital twins, i.e., digital representations of physical objects, whose value can be expressed in financial terms. In addition, blockchain facilitates the secure transfer of digital assets by means of asymmetric cryptography. The generation of the public-private key pair enables users to create certificates that prove assets' ownership. Once the transaction has been validated by the network and stored on blockchain, the new owner has control over the asset. Ultimately, the IoV concept entails empowering the final user who embodies a set of roles, such as creator, consumer, certifier, owner and protector of information,

instead of the limited roles of producer and consumer in the Internet of Information. This means that the crucial part of the system should be digital assets that are protected by their legitimate owners using their wallets. On contrary, the center of blockchain ecosystem is the wallet, whereas the digital asset is usually stored off-chain, due to performance efficiency and scalability limitations of blockchain platforms.

Neural distributed ledgers (NDL) are technological platforms inspired by blockchain promises and unique characteristics of human brains [65]. These platforms aim to address the aforementioned issues of conventional blockchain platforms, such as interoperability, performance and scalability, and ultimately enable the global implementation of the IoV paradigm. The first academic paper that discussed neural distributed ledgers and their similarities with neurons aggregation, due to interconnected subsets of groups that work in a parallel way was published in 2020 by Velasco et al. [65]. In fact, the authors of the NDL study [65] were inspired by the novel idea of Swan [54] that entails developing blockchains as “personal thinking chains”. Fundamentally, the neural blockchain is internally organized into subsets of groups that perform work in parallel and are interconnected similarly to how neuron groups are interconnected in human brains. Recently, the concept of NDL has been adopted in a few studies, for instance, Benítez-Martínez et al. [66] designed a NDL-enabled e-Participation model that uses virtual tokens to incentivize greater participation of citizens in public affairs, and in a later study [67], the same authors used neural distributed technology to hold public procurement related information in blocks in a transparent and secure manner to avoid corruption.

Recently, ArcaNet has been proposed as a collaborative P2P platform that enables the protection and end-to-end secure transfer of all types of digital assets [68]. This platform relies on the concept of NDL Arca as a distributed token repository that ensures the protection of tokenized information against illegitimate access [68, 69]. Francisco Luis et al. [69] referred to NDL Arca as a virtual safe box that uses two private keys to access the content of the tokens. Tokens are created as a result of the tokenization process that encompasses transforming a tangible or intangible object into a unique, exclusive and valuable set of data. There are a set of requirements that should be fulfilled for the data set to be considered a token:

- identifiable,
- has a recognizable creator and owner who is able to modify the content of the token,
- has a market value associated with its evolution,
- original (ensuring originality requires digital signatures from creators, trusted and random certifiers),

2. Background

- able to enter and exit markets based on owners' interests.

Due to their value, it is necessary to store tokens in a secure store throughout their entire lifecycle: emission, evolution, transfer and destruction of tokens, that prevents illegitimate access and unauthorized modifications. NDL Arca aims to achieve this by grouping tokens into tables and tables into databases in a hybrid format between key-value storage and relational databases. The keys are expressed in ULID (Universally Unique Lexicographically Sortable Identifier) format to identify contents in any environment. The ULID format generates identifiers as a result of a combination of timestamp (first 10 characters) and randomness (remaining 16 characters). The associated values refer to dynamic tables (variable array []) and are always encrypted. The columns of the tables consist of token fields' ID and fields' content. It is worthy to outline the dynamic nature of these fields which allows users to change them according to their needs. Furthermore, each token consists of a public part which is encrypted with the database key in order to be accessed only by valid users of NDL Arca, and a private part which is encrypted with the private key of the token and can be read and updated exclusively by the owner of the token. Tokens' fields are mutable and configurable and can entail diverse content, such as large files, complex objects, serialized Java objects, digital signatures of other fields, checksum controls, library .jar, or bpmn executable process.

NDL Arca adopts a set of techniques to ensure its security, such as double-key encryption, as the data is encrypted with a database key and token key, AES 256 (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman), hashing functions and zero trust [7]. While NDL Arca can work in standalone mode to create a centralized dedicated server or database system in the cloud, its primary use is in blockchain networks, due to its inherent ability to replicate, perform hashing of contents and distribute them across the P2P network. NDL ArcaNet refers to the use of Arca in a multi-domain network. In NDL ArcaNet, users act as the protectors of their own tokens hosting them in their own wallets and set their own exchange rules for assets. The transfer of assets is done directly between parties without any type of mediation. Actions applied on tokens will be checked by a pseudo-random set of network users. The users that certify actions on tokens by signing digitally receive a commission as service payment.

It is worthy to mention that the certifiers sign the actions performed on tokens without having access to the content of the assets. Each token generates internally its own private signed blockchain ledger to certify its provenance, content integrity, evolution and history. As mentioned previously, the fields of these tokens can be editable and store any type of value ranging from simple to large files or executable processes. This platform can be used for certification of innovative entities, professional capabilities, industrial processes (IoT), industrial secrets (patents), NDA

(non-disclosure agreement) for trading industrial secrets, encrypted P2P mail communications, and e-commerce.

NDL ArcaNet has a set of advantages that are relevant in the software engineering context, as follows:

- *real decentralization.* In NDL ArcaNet, each wallet safeguards its assets and applies its own governance rules. Assets certify their own state and history using an internal signed blockchain. Therefore, each asset can be considered a portable blockchain data structure that has its contents signed by a set of at least 3 trusted and 5 random validator wallets. Certifier or validator nodes receive a fixed payment for each signature provided by their wallets.
- *performance efficiency.* Transactions are verified and applied by each wallet independently, enabling real-time and parallel work and consequently maximizing the number of transactions per second (tx/s). In fact, each wallet can process over 300'000 tx/s. Since in NDL ArcaNet, the digital asset is unique and is hosted and managed by a single wallet at a time, the system does not need a global consensus to synchronize a global data set in order to function properly. Instead of that, the platform enables a pure P2P data storage platform where each wallet is an independent actor with full control and governance over its own hosted dataset, hence it decides what data to expose to other wallets in a certified fashion. If different entities perform write operations on the same content, an internal semaphore is used to solve synchronicity issues. Moreover, the lack of consensus means that each wallet functions as a local database but with higher latency because of signature mechanisms. This implies that limitations related to real-time operations in NDL ArcaNet are comparable to centralized databases limitations.
- *scalability.* NDL ArcaNet is designed to integrate millions of nodes or wallets because they work independently and in parallel. Although NDL ArcaNet transfers signed data packets, it is able to scale similarly to the Internet [7].
- *sustainability.* The platform is based on collaboration, rather than competitive-based and resource-wasteful consensus algorithms, such as PoW or PoS, and does not require gas to carry out transactions.

Finally, NDL ArcaNet ensures immutability by creating a map <ULID, keypair> that stores all valid signatures for each wallet, and deleting the private key of the keypair after the first use by default. Private keys are deleted to prevent the wallets from using the same signature twice. Each block is signed by wallets that use the keypair associated to the ULID specified by the previous block randomly. Interestingly, each block in NDL ArcaNet protects the previous block by including the signed hash of the previous block (the past) and assigns <validator, ULID> tuples to protect

2. Background

the next block (the future). By doing so, blocks cannot be modified and re-signed in an illegitimate way.

2.2 Software Engineering

“Software engineering is the part of computer science that is too difficult for the computer scientist”.
(Friedrich Bauer)

During the last 5 decades, software has evolved from a technological tool used to address specific problems, into an industry that has become ubiquitous in every corner of today's society [70]. The development of high-quality software requires following a software process tailored to specific business needs. The software process, along with technical methods and tools used has been referred to as software engineering. One of the earliest definitions of software engineering (SE) was presented in the first NATO conference in 1968 [71]: *“software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines”*. A more comprehensive definition of SE was provided by IEEE Standard 610.12 [72]: *“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 software engineering field has been guided by the Software Engineering Body of Knowledge (SWEBOK) that defined the following knowledge areas [73, 74]: software requirements, software process, software testing, software quality, software maintenance, software configuration management and engineering management, just to mention a few. In what follows, the global software engineering paradigm and requirements engineering field are introduced. Furthermore, the main focus of the section lies in defining RT, and presenting RT technologies and challenges.

2.2.1 Global Software Engineering

Today's software products are designed, developed, produced, and maintained as a result of complex, global supply chains that consists of a variety of distributed partners in each of the phases of the software lifecycle [75]. This new paradigm has been referred to as global software engineering (GSE) [75]. The increasing interest, particularly in developed countries regarding GSE is not surprising, due to tangible economic benefits [76]. Client organizations leverage the lower costs of vendors in developing countries, compared with in-house development, and in addition they leverage the 24-hour development model [76]. However, the complex nature of GSE projects gives rise to both client and vendor related challenges. Niazi et al. [76] conducted a systematic literature review that aimed to identify client and vendor challenges from a research perspective, followed by performing questionnaires that targeted both client and vendor organizations. The analysis did not reveal significant discrepancies between neither academia-industry, nor client-vendor organizations. However, the authors identified 19

challenges, such as lack of communication, lack of cultural understanding, lack of knowledge transfer among teams, lack of a uniform process among teams, lack of trust, risk management and requirements engineering challenges, among others. It is not surprising that geographical distance is likely to lead to the lack of project visibility, delays and requirements elicitation problems [77].

The aforementioned GSE challenges have been mitigated to some extent by adopting agile practices, such as sprint planning and stand-up meetings, customer collaboration and retrospectives [78, 79]. While it is true that these practices enhance communication, collaboration, transparency, visibility, and consequently trust among teams, it is also true that agile practices were not designed with GSE in mind [80]. In fact, it has been reported that scaling up agile practices introduces coordination challenges among teams, due to their enhanced autonomy [81]. The enhanced autonomy may in turn cause technical discrepancies, e.g., in coding style and consequently, it can deteriorate trust among teams [81]. A deeper focus on coordination challenges in large-scale agile development and how to address them is outside the scope of this PhD project, but the interested reader is referred to a recent case study on the topic performed by Berntzen et al. [82].

2.2.2 Requirements Engineering

Requirements engineering (RE) is an important activity of the broader SE field, as it impacts in a significant manner the effectiveness of all the subsequent phases of the software development lifecycle [83, 84]. RE covers a set of intertwined activities entailing the elicitation, analysis and specification of requirements that reflect the intended goal of a software system by taking into consideration and aligning the perspectives of all relevant stakeholders [85]. The significance of RE has been acknowledged over the last three decades by the SE community. In 1983, Boehm, as cited in [83], estimated that fixing errors in system requirements can be 100 times more costly compared with errors introduced during the later phase of system implementation. In 1993, Lutz [86] investigated software errors in NASA's Voyager and Galileo spacecraft programs and found out that the majority of safety faults were caused by functional and interface requirements errors.

Likewise, in 2002 Hall et al. [87] performed a case study and their findings revealed that almost half of the development problems were requirements-related problems. A more recent initiative was started by RE researchers to explore the status quo and challenges in practical RE and it has been referred to as NaPiRE (Naming the Pain in Requirements Engineering) [85, 88, 89]. NaPiRE consisted of a set of globally distributed surveys targeted at different companies in a variety of domains. The top 3 RE-related challenges identified and elaborated in the paper published in 2017 [85] entailed incomplete or hidden requirements, communication flaws between the customer and the team, and moving targets. These challenges are not surprising, given the inherent complexity of the RE discipline which makes the standardization of the field challenging [85]. This complexity is caused by the interdisciplinary nature of the field, dependency on customers and a

2. Background

variety of other stakeholders, and uncertainty [85]. Uncertainties trigger changes in requirements, and managing such changes is not trivial, particularly in global software engineering [90].

While RE is already challenging in co-located development environments, it is even more challenging when requirements are specified and often changed by cross-organizational stakeholders across cultural, geographical, language and time-zone boundaries [91, 92]. The complexities, that such a global environment brings, may explain why theoretically important RE activities, such as RT are not implemented in practice or are implemented in an ad-hoc fashion [14, 17, 27, 31, 93]. The following sections aim to shed light into RT as a process and associated challenges.

2.2.3 Requirements Traceability: Definition and Technologies

RT is not a new concept in RE, in fact it has been introduced over 30 years ago [94]. The accurate and reliable creation of trace links is important to support a variety of software development lifecycle (SDLC) activities, such as change management [18, 95], software maintenance [96, 97], and project management [18]. Traceability is particularly important in safety-critical systems, such as finance, automotive, aerospace, and healthcare, as the inadequate use of traceability may lead to major financial loss or life-threatening accidents [14]. Hence, in such domains, traceability is mandated by several software process improvement and capability models and standards, such as CMMI (Capability Maturity Model Integration), ISO (International Organization for Standardization) 26262, and ASPICE (Automotive Software Performance Improvement and Capability dEtermination). Although the RT definition provided by Gotel and Finkelstein [11] has been commonly accepted as the standard definition of RT, other definitions that vary in purpose and scope have been proposed [31]. This PhD dissertation adopted the following two definitions:

“ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of ongoing refinement and iteration in any of these phases)” by Gotel and Finkelstein [11].

“the ability to relate artifacts created during the development of a software system to describe the system from different perspectives and levels of abstraction with each other, the stakeholders that have contributed to the creation of the artefacts, and the rationale that explains the form of the artifacts” by Spanoudakis and Zisman [98].

While the first definition focuses on the lifecycle of requirements, the second definition is broader and encompasses all software artifacts. Other definitions relevant to the topic are those of vertical and horizontal traceability. While vertical traceability refers to tracing artifacts at different levels of abstraction, e.g. requirements and code, horizontal traceability refers to tracing artifacts at the same abstraction level, e.g. traces between all requirements specified by a specific stakeholder [10]. A more logical categorization is that

of pre-requirements traceability (requirement's life before being included in the requirements specification document) and post-requirements traceability (requirement's life from the inclusion in the requirements specification document) [11]. Particular attention needs to be paid to pre-requirements traceability, given that it has been identified as the main contributor to the requirements traceability problem [11]. Moreover, the requirements traceability process consists of three main phases at the individual trace level: creation, maintenance and usage [10], which will be referred to throughout this paper. These activities can be done manually, semi-automatically or full-automatically [99]. Trace links have been traditionally created as a result of a time-consuming and tedious manual process which often leads to inconsistencies, in particular in large and complex projects with a variety of stakeholders, tools and artifacts. On the other side, due to the fact that full-automation is not reliable and accepted by practitioners [100], a semi-automatic tracing process is employed in most cases. Candidate traceability links (so called because they have to go through a vetting process in order to be considered final) are generated from the tools and then, they are examined by a human analyst who decides whether to accept or reject the candidate link.

A vast number of technologies and tools have been proposed by the RE and traceability community. Wang et al. [20] provided a comprehensive overview of the main technologies which were categorized into the following groups:

- *RT generation technology.* Generating trace links reliably is a trivial task due to the diversity of evolving artifacts that need to be traced to and from requirements, at different granularity levels. Therefore, it is not surprising that proposing new automated generation methods and improving existing generation methods has been the main research focus of the traceability community. Wang et al. [20] presented a set of generation methods, however in this thesis only two of them are explained, as automated generation methods are outside the scope. These generation methods are described, as follows:
 - *Information retrieval (IR)-based tracing methods.* IR-based methods are the most popular generation methods, due to the textual nature of requirements documents and their associated artifacts [20]. According to Wang et al. [20], the most commonly used IR tracing methods are Vector Space Model (VSM), Latent Semantic Indexing (LSI), and Probability Model (PM). The main idea of such methods lies in the textual similarity comparison between source and target artifacts. These artifacts are more likely to be related in case of higher textual similarity. However, the terminology can be missing, duplicated or inconsistent which may affect the performance of IR methods in terms of precision and recall.
To address such issues, a set of strategies have been proposed, e.g., the use of thesaurus, project glossary and refactoring.
 - *Data mining and machine learning methods.* These methods rely on the reuse of information, knowledge or patterns that are extracted

2. Background

from pre-existing trace links, thus it is not surprising that they often outperform IR-based methods. Data mining algorithms extract trace links patterns from pre-existing artifact pairs, for instance affinity mining provides information, e.g., domain glossaries, and associations between patterns based on term-occurrence and trace links, from pre-existing requirements links. This information is then used to generate new trace links. Moreover, machine-learning methods extract information from pre-existing trace links, build a linking patterns model based on the extracted information and use the model to suggest candidate trace links. The quality of the output in such methods depends significantly on the quality of the training set. Other tracing generation methods can be found in the systematic literature review carried out by Wang et al. [20].

- *RT refinement technology.* Trace links generated by automatic tracing methods are often not accurate. Hence, three main refinement techniques have been proposed: (i) manual methods, such as relevance feedback and tagging. Relevance feedback methods rely on an analyst to annotate (a tagging interface can be used) candidate trace links as true or false, and then use the annotation to refine the generation model and improve the accuracy of trace links. (ii) methods based on structural information. These methods use information about artifacts' structure and features to refine candidate links. For instance, trace links between requirements and source codes written in object-oriented programming language can be refined by clustering classes on the basis on inheritance. This means that classes can be assigned to the same cluster as parent classes. These methods can enhance the recall rate of candidate trace links, assuming that classes in the same cluster have similar probability to generate trace links from source codes to requirements. (iii) hybrid methods are methods to refine trace links generated from different generation methods or information sources. For instance, four information sources, i.e., class names, comments, method names and variables can be used to generate trace links from source codes to requirements. Candidate trace links generated by more than two sources are considered final trace links, as determined by the voting algorithm.
- *RT maintenance technology.* The quality and accuracy of trace links tend to deteriorate, as the system evolves. Thus, trace links must be maintained in an automatic fashion, and this can be achieved by means of: (i) rule-based methods that rely on analysing change patterns in source and target artifacts, creating mapping rules between the two types of change patterns, and updating trace links automatically by using predefined rules, when changes occur. (ii) scenario-based formalization method. Formalized scenarios can be used as intermediary artifacts between requirements and test cases. These intermediary artifacts reduce the human effort of maintaining trace links between requirements and test cases. The reason lies in the fact that test cases can be derived from formalized scenarios in an automatic fashion, and consequently fewer trace links have to be

maintained between requirements and formalized scenarios compared to requirements and test cases. (iii) general-purpose ontology. Defining a traceability ontology facilitates semantic annotation and reasoning in both source and destination artifacts.

- *NFR-tracing technology.* Cleland-Huang [101] outlined that tracing non-functional requirements (NFRs) is significantly more difficult compared with functional requirements due to their multi-dimensional impacts across the software system, interdependencies and tradeoffs between NFRs and software architecture. Hence, conventional traceability methods are insufficient to trace NFRs [101]. Three main methods have been proposed in this regard: (i) aspect weaving method that models and implements NFRs as aspects and traces them to codes and tests, (ii) design patterns methods that rely on the use of pattern detection algorithms to trace NFRs to system design, and (iii) IR-enabled methods that are based on the underlying assumption of text similarity between source and target artifacts. Thus, Text Semantic Similarity (TSS) can be used to generate relations between NFRs and code classes.
- *RT representation technology.* These technologies enhance the understanding of trace links, and consequently encourage the application of such links. The most common representation method is the Requirements Traceability Matrix (RTM), however other methods have been proposed to enhance the representation content. In particular, these methods focus on extending tracing objects, e.g., extending the RTM to include tracing relations from source artifacts to developers, and enriching trace links types. Instead of just saying if there is a relation between artifacts, other methods can be used to illustrate the confidence degree of each trace link, for instance marks, colors and numbers. In addition to the confidence degree, the relation type can be defined by means of a RTM ontology. In addition, the methods focus on incorporating trace link states. The continuous changes of requirements trigger changes of trace links states. These changes of trace links states can be recorded by importing UML state diagrams. Furthermore, the representation form can be enhanced with the use of ordered relevance links that sort trace links based on their confidence degree, and trace graphs. Trace graphs use nodes that differ in terms of color, size, and shape to represent entities of different significance and type, and edges that differ in thickness to represent traces and their confidence levels.

2.2.4 Requirements Traceability Challenges

Despite the theoretical benefits of traceability, in practice traceability is not implemented at all or implemented in an ad-hoc manner [14, 95]. This has motivated the traceability community to investigate challenges of implementing RT in practice. In this regard, the foundational work was carried out in 1994 by Gotel and Finkelstein [11]. The results of their study suggested that the lack of a unanimous definition and the expectation that traceability can address

2. Background

multidimensional issues in a variety of projects, users and tasks, often conflicting, contributes to poor RT practices. In 1998, Ramesh [12] carried out surveys in 26 organizations that differed in terms of motivation to implement traceability. Thus, the organizations were categorized into two main groups: (i) low-end users of traceability implement traceability because of regulatory requirements, and (ii) high-end users of traceability implement traceability to enhance the quality of their software systems. This was one of the first studies that explored factors influencing traceability practice from different perspectives: environmental, organizational and system development.

Blaauboer et al. [21] performed a case study in a large IT consultancy company to identify the dominant factors that impact the decision on whether to adopt traceability in information systems development projects. First, they considered 8 project managers in the company since their initial verification revealed that project managers are responsible for making RT-related decisions in the company. Interestingly, only 3 out of 8 managers were familiar with the term “traceability”. Finally, the authors selected 6 managers who were aware of traceability and conducted in-depth interviews with them. The results suggested that the most relevant factors for making a decision about RT are development organization and customer awareness, return on investment, stakeholders’ preferences and process flow. In addition, the results indicated that the prominent reasons for not adopting traceability are the lack of awareness and limited incentives to adopt traceability.

Winkler and Pilgrim [31] conducted a survey to enhance the theoretical and practical knowledge on traceability in both RE and model-driven development areas. The authors shed light on factors limiting the application of traceability in industry. These factors were categorized and summarized, as follows:

(i) natural. The lack of a commonly accepted complete traceability scheme limits its application in industry. In fact, the complete capture of traces is not trivial, due to the imprecise nature of human work methods and the incorporation of social and implicit knowledge. (ii) technical. Technical limitations exist even if a well-defined traceability scheme is used. The intervention of humans is necessary to identify semantics of information and record only relevant traces that are meaningful in a specific context. Technical limitations, such as lack of tool integration and poor tool support for maintenance were also outlined. (iii) economical. According to the authors, project managers tend to neglect traceability because of the lack of convincing evidence of traceability benefits at a project or company level. (iv) social. While management support for traceability is necessary, its proper implementation depends on the staff and their personal motivation. Their low motivation is likely to affect the quality of traces negatively.

Nair et al. [102] carried out a systematic literature review on 70 primary studies published within 1993-2012 in the proceedings of the International RE conference. The analysis of these studies revealed that the most dominant challenges addressed were the lack of knowledge and understanding of traceability and traceability maintenance when requirements evolve. In addition, the authors of the study recommended further research efforts into the following

topics: traceability visualization, consideration of software artifacts other than requirements, trace semantics for impact analysis and advanced empirical evaluation. Two more recent studies on the topic are those of Wang et al. [20] and Maro et al. [14]. The systematic literature review conducted by Wang et al. [20] pointed out trustworthiness and automation as the most frequent challenges addressed by RT technologies. This is not surprising, given the importance of the RT generation activity, but also its complexity and fallibility for both (semi-)automated methods and analysts.

Maro et al. [14] carried out a tertiary literature review to identify general domain-agnostic challenges of traceability, and specific traceability-related challenges in the automotive domain. Differently from Wang et al. [20] that focused on challenges addressed by RT technologies, e.g., generation technologies, maintenance technologies, and representation technologies, Maro et al. [14] expanded the scope by encompassing RT activities, such as preparation and planning, establishment, outcome, and exchange of traceability information. A set of 22 challenges were identified by the tertiary review of Maro et al. [14], and the majority of them were confirmed by the case study in the automotive company. An interesting finding of the tertiary study is that employees responsible for creating trace links fear the misuse of this data against them as a means to judge their performance. Moreover, the findings indicated that traceability is perceived as overhead by developers who are often not the users of the trace links they create. This demotivates developers to create trace links, resulting in missing or even wrong links.

Other relevant studies on the topic can be found in a previous paper written and published within the scope of this PhD project [103].

2.3 Blockchain and Software Engineering

2.3.1 Software Engineering for Blockchain Technology

Recently, researchers such as Porru et al. [104] explored SE aspects in blockchain-based software implementations and coined the field blockchain-oriented software engineering (BOSE). Porru et al. [104] reported that software engineers perceive blockchain-based software projects as being developed in an unruly and hurried manner, without taking into consideration SE concepts. Therefore, software quality is not ensured in such projects. Due to the safety-critical nature of blockchain applications, it is necessary to use reliability and security methodologies, such as Cleanroom SE, software reviews, smart contracts testing, and blockchain transaction testing. Furthermore, the distributed nature of blockchain calls for the modification of existing UML (Unified Modeling Language) diagrams or the creation of new ones, and the introduction of new metrics to measure resource consumption, complexity, and performance of blockchain-enabled systems [104]. Finally, to enhance the quality of blockchain-oriented software projects, Porru et al. [104] outlined the need for professionals with skills in different dimensions beyond technology expertise, for instance skills in finance and law.

2. Background

The need for BOSE discipline was also advocated by Destefanis et al. [105]. The authors analysed the Parity Wallet hack on Ethereum in 2017 that resulted in the freezing of approximately 500K Ethers with a single library code deletion. The analysis of the source code of the library where the bug was discovered suggested that the library vulnerability was mainly a result of negligent programming, instead of problems in the Solidity language. In fact, the exploitation of the vulnerability was easy and performed in only two steps: (i) The smart contract (SC) library was left uninitialized, allowing the attacker to become its owner by calling the initialization function `initWallet()`. (ii) The owner of the library contract is now able to call privileged functions, e.g., `kill()` which in turn calls `suicide()`. The latter sends the remaining part of funds to the owner and then, clears the storage and code of the contract. In this case, the library contract was killed and this inevitably affected all the SCs that relied on calling the library to execute functionality. According to the authors, the effects of this (involuntary) attack could have been mitigated, should the following best practices had been adopted: (i) defining design patterns, i.e., anti-patterns and patterns. An example of an anti-pattern can be creating SC library and leaving it uninitialized, (ii) proper testing techniques. The immutable nature of SCs once they are deployed on blockchain does not allow for testing, therefore robust testing techniques must be adopted on testnets, before deploying SCs. The authors suggested a combination of manual and automation testing, e.g., state-based MBT (model-based testing) techniques.

Recently, Vacca et al. [32] analysed 96 studies published from 2016 to 2020 that focused on SE issues of smart contracts and blockchain development, e.g., SC code analysis, security and testing. The results revealed that the existing literature has paid significant attention to SC testing in terms of finding bugs and vulnerabilities, but other types of tests, such as integration and regression have been overlooked. Although SC vulnerabilities have been identified, further efforts can be devoted to creating a reference taxonomy that uses a set of criteria, e.g., scope, impact, cost and effect, to organize vulnerabilities. In addition, the authors suggested the creation of a catalog of attack detection patterns related to SCs, and a systematic methodology (including practices, tools and procedures) for SC security testing.

2.3.2 Blockchain Technology for Software Engineering

While blockchain technology has been implemented and adopted in different industries [106, 107], its application in the SE domain has not received enough attention. Although, the cross-fertilization between blockchain technology and SE was advocated recently by SE researchers, such as Marchesi [108] and Colomo-Palacios [109], limited efforts have been devoted to the development and implementation of blockchain solutions to address SE issues [110]. In what follows, the main research efforts are presented.

Tariq and Colomo-Palacios [33] published the first systematic mapping on the topic in 2019. Despite the relatively low number of studies reviewed, the authors provided interesting results that can trigger further exploration and

investigation of the topic. According to the authors, a blockchain system based on smart contracts can be valuable in establishing a trustworthy relationship between parties that do not always trust each other, e.g., developer and employer in outsourced software development. In addition, a blockchain-based system can be used to track developers when adding third-party components to the end product, and automatically check the adherence of these components to license compliance policies. Adding third-party components without reporting to the other team parties can have a negative impact on the quality of the software and the reputation of the organization. This can be mitigated with enhanced visibility and auditability, as inherent properties of blockchain systems. Moreover, smart contracts can be applied in cyclometric complexity to predict the complexity of code, prior to the development phase. Finally, the authors outlined the need for professionals with skills in both blockchain technologies and SE, to further advance the blockchain-oriented SE knowledge and competence.

In 2019, Beller and Hejderup [111] advocated the use of blockchain to address SE issues, and coined the term blockchain-based software engineering. The authors proposed the following two blockchain systems: (i) a blockchain-continuous integration (CI) system which intends to prevent single point of failures that occur in traditional CI systems. In addition, the system establishes an equal market for computing power which in turn regulates build prices according to the demand-supply market rules. (ii) a blockchain-enabled package repository that enables anyone to propose new packages and assess the work of others. This approach empowers library users, as it enables a wider part of the community to contribute to the proper release of packages. According to the authors, blockchain can contribute to the SE landscape in three dimensions: (i) professionalization by rewarding those who participate in proposing or verifying builds or packages, instead of relying in the work of volunteers as it occurs in open-source software. (ii) enhances software artifacts quality, due to the verification of builds and releases. (iii) trust and availability, as opposed to centralized systems, e.g., GitHub or Travis CI.

Other authors proposed blockchain-based solutions to address issues in distributed and cross-organizational software development environments. Krol et al. [112] presented a blockchain-based platform for outsourcing software development, namely ChainSoft. This platform enables software requestors to publish tasks, along with tests to verify the compliance of code with requirements. In addition, the platform enables developers to create and submit code which is verified automatically by means of smart contracts, and consequently payment is delivered to the developer. An important component of the proposed platform is Oracle that entails a smart contract that communicates with GitHub/Travis CI. The reliance on GitHub/Travis CI may introduce two main risks: (i) their dishonest behavior can lead to running tests incorrectly, (ii) unavailability of their services can cause the resubmission of the task or solution by the requestor or developer, respectively, with an additional transaction fee. Yau and Patel [29] addressed communication and coordination issues among development teams which are often distributed and are required to collaborate for the development of complex and large-scale software systems. Their proposed approach enables

2. Background

software teams to generate software specifications for each of the software development lifecycle phases in the form of smart contracts. These smart contracts contain information about allowed teams, information useful to carry out software activity and acceptance criteria in key, value format, and terms for the acceptance or rejection of the software output.

Yilmaz [113] integrated blockchain with large-scale software development to foster trust and integrity. The authors proposed a conceptual model enabled by blockchain technology that aims to decentralize the test-driven software production line by using the consensus concepts of “Proof-of-Work” and “Proof-of-Stake”. According to this model, the leader publishes new works to the network consisting of the description of work, description of test cases, and rewards. Developers act as miners, fulfil the work, and publish their code, whereas testers act as validators, assess developers’ works, and publish candidate blocks which are validated by a Proof-of-Stake consensus algorithm, and then merged to generate consequent blocks. The authors reported on the loss of trust between parties that pushes software practitioners to find solutions on the working software process or call for third-party mediation. A software development process based on blockchain can offer guidance to solve such trust issues.

Lenarduzzi et al. [114] used blockchain and smart contracts to manage Agile projects that are based on Scrum or Lean-Kanban methodologies. Their proposed model starts with the customer who creates a SC for a specific project, specifies the product owner (PO) address, and the amount of Ethers for payment. The PO registers user stories, acceptance tests, the hash of the expected result, state, and developer’s address. Developers submit to the SC the hash digest of the acceptance tests’ result which is then checked against the hash of the expected result. Once the result is verified, Ethers are sent automatically to the developer’s address. This model aimed to facilitate the duties of the PO, due to the contribution of SCs in certifying the correctness of the results automatically. A similar approach has been followed by Farooq et al. [115] with their AgilePlus framework that aimed to address trust, communication and coordination issues between parties in distributed agile development. While this framework executed smart contracts for reliable payments and acceptance testing verification, similarly to Lenarduzzi et al. [114]’s approach, it takes a more inclusive view of the overall agile development process. AgilePlus consists of six abstraction layers that are aligned with the agile development process and each layer is only activated when the previous process is terminated by either the customer or developer. These layers are agreement, requirements elicitation, prioritization, design and development, testing, and payment. Due to the performance evaluation of their framework, the authors strongly believe that blockchain has the potential to disrupt distributed agile development.

A group of authors from Accenture published a series of papers that use blockchain as the backbone of the software development process [116, 117, 118, 119, 120]. The authors presented three main blockchain-enabled frameworks:

- (i) Blinker framework for trusted software provenance [118]. This framework

uses data ingestion tools/plugins to capture provenance data from the multitude of tools adopted throughout the software lifecycle. These data are then transformed according to PROV-specifications, and their validity is approved by participants selected in accordance to voting policies. To enhance the understanding of provenance data, the framework allows agent-, artifact-, and process-centred queries to be performed, and enables interactive hierarchical representation of provenance data.

- (ii) a software identity framework, namely ShIFt that derives sub-identities from software components, such as code or third-party components [119]. The combination of these sub-identities forms the software composite identity which is stored on blockchain ledger. This ensures software integrity, as it is possible to identify integrity discrepancies between two software instances, and their causes.
- (iii) a tokenized incentive framework that uses smart contracts to analyse event data against incentive policies [120]. In addition, smart contracts deliver incentives to software engineers for the performed tasks, in the form of wallet tokens, accordingly. This approach ensures the transparent delivery of outcome-based and eternal incentives to participants that contribute to the development of complex software systems in a globally distributed fashion.

Although these studies contribute to the emerging blockchain-based software engineering field, they do not address the requirements engineering and traceability processes in interorganizational software projects. In fact, ensuring trustworthy requirements traceability is a complex topic, due to a variety of tools used throughout the software development lifecycle, a variety of distributed stakeholders, and a high number of ever-changing software artifacts. To the best of our knowledge, this thesis is the first research effort devoted to exploring the use of blockchain technology for trustworthy requirements traceability, to date.

Chapter 3

Research Methodology

*“Research is an organized method for keeping you reasonably
dissatisfied with what you have”.*

(Charles Kettering)

3.1 Design Science Methodology

The research methodology adopted in this dissertation is presented in Figure 3.1. This methodology was adapted from the design science research methodology proposed in the seminal paper of Hevner et al. [2]. According to these authors, design science relies on building and evaluating artifacts with the goal of achieving utility. The evaluation of artifacts can lead to the identification of weaknesses and consequently, the need to refine the artifacts. The build-evaluation phase makes use of the raw materials provided by the knowledge base component that is composed of frameworks, theories, models, and methodologies. The appropriate application of research foundations and methodologies ensures rigor [2]. It is worthy to outline that design science differs from routine design in that it proposes innovative solutions to address existing relevant problems, hence contributing significantly to the existing knowledge base.

The papers published within the scope of this PhD dissertation follow different building blocks of the research methodology presented in Figure 3.1. The first building block of research foundations was executed during the first year of the PhD project. During the first year, the PhD student acquired relevant foundational knowledge related to the following topics: (i) blockchain technology concepts (e.g., inherent features, blockchain platforms, and blockchain applications), (ii) distributed and global SE, (iii) RT challenges and technologies proposed to address the challenges, and (iv) bidirectional relationship between blockchain technology and SE, with a particular emphasis on the application of blockchain to address relevant SE issues. In addition to technical knowledge, the PhD student acquired knowledge regarding research methodologies, e.g., how to conduct systematic literature reviews and systematic mapping studies, how to collect data by means of semi-structured interviews and how to analyse data through grounded theory and content analysis techniques. To acquire this knowledge, the PhD student participated in blockchain, SE, and qualitative research methodology related courses at UiO, read relevant scientific literature and participated in conferences. As a result of this process, three scientific papers were published during this first phase (P1, P2, P3).

The acquired knowledge was applied to build the initial blockchain-enabled framework for RT. The related paper was published in EuroSPI 2021 conference (P4). The initial framework was then refined in a more practical way by carrying

3. Research Methodology

out semi-structured interviews with blockchain experts. The related paper was published in the Journal of Software: Evolution and Process (P5). Based on the updated framework, a prototype named BC4RT (Blockchain for Requirements Traceability) was developed and explained in another paper which was presented at EuroSPI 2022 conference (P6). Finally, the proposed framework and BC4RT prototype were evaluated by means of SE experts who provided the researcher with interesting insights on how to further enhance the quality of the proposals, that led to promising future research directions (See Chapter 5). The last paper (P7) has been submitted to the Journal of Software: Evolution and Process.

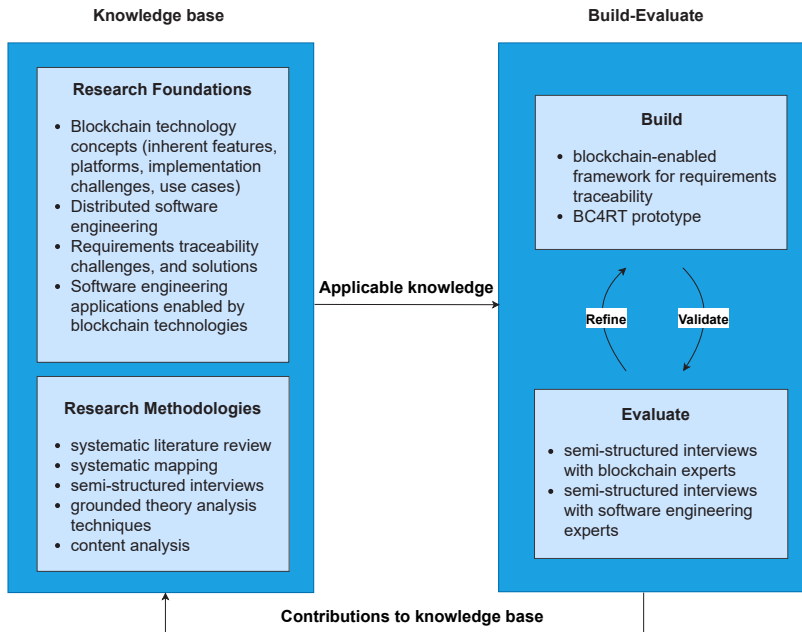


Figure 3.1: Research methodology, adapted from Hevner et al. [2]

3.2 Research Design: A Multi-Method Approach

The underlying philosophical paradigm of this thesis is *pragmatism*, because it encourages researchers to carry out empirical inquiries in order to address real-world practical issues, while acknowledging the existence of singular or multiple realities [121]. Hence, it is common for pragmatist researchers to adopt a pluralist research design strategy which in this thesis refers to the use of multiple qualitative data collection and analysis methods. The justification for applying a qualitative multi-methods strategy was motivated by the underlying paradigm,

the exploratory nature of this thesis and the research questions formulated in Section 1.2. In what follows, the methods are presented in relation to the research questions.

RQ1 aims to explore the challenges of implementing RT, as reported by recent literature. Thus, a systematic literature review was considered the most suitable approach to address this research question. RQ2 aims to explore blockchain use cases in the SE field, hence a systematic mapping approach was adopted to map blockchain properties with SE challenges. Finally, RQ3 is addressed by designing, refining, implementing, and evaluating a blockchain-enabled framework that can be used by organizations to enable decentralized and trustworthy RT in interorganizational software projects. Since the framework should be used by organizations, its refinement aimed to enhance the practicality. Therefore, blockchain experts' judgment was incorporated by means of semi-structured interviews and grounded theory analysis techniques. While the interview-based qualitative approach was adopted to collect rich and in-depth data from experts in the field, grounded theory analysis techniques were adopted, due to their soundness and the novelty of the blockchain-enabled RT topic. Finally, an interview-based qualitative approach was followed to evaluate the strengths and limitations of the proposed framework and prototype with SE experts. In this case, the data was analysed by means of content analysis technique, due to its ability to emphasize the content of words. These methods are explained in detail in the following sections.

3.3 Data Collection

3.3.1 Systematic Literature Review

A systematic literature review was carried out in the initial phase of this PhD project. This SLR relied on the well-known and sound guidelines/procedures for conducting literature reviews in SE proposed by Kitchenham [36, 122, 123]. The process consisted of three main phases that are presented in the following section. For a more detailed explanation, the interested reader is referred to the second thesis paper (P2). The three phases are described, as follows:

(i) *Planning the review.*

In this phase, the review protocol was developed as a result of brainstorming sessions with two other researchers. Each brainstorming session provided feedback that contributed to enhancing the initial protocol. The protocol consisted of three sections:

- Development of a research question (RQ) that adheres to the structure (population, intervention, outcome) suggested by Kitchenham [36].
- Selection of a search strategy. Database search was the main search strategy adopted in this dissertation because it has been the recommended search approach in SE [124]. However, given the noise generated by database searches, backward snowballing

3. Research Methodology

technique which entails scanning reference lists of selected studies, was adopted as complementary to databases search. In addition, a search string that incorporated the relevant terms according to the RQ was formulated and executed in 5 major databases: IEEE Xplore, ACM Digital Library, Springer Link, Science Direct and Wiley Online, as recommended by Kuhrmann et al. [125].

- Given that database searches generate a vast amount of studies, it is necessary to analyze the retrieved studies against inclusion and exclusion criteria.
- (ii) *Conducting the review.* This phase consisted of selecting studies and assessing their quality. Figure 3.2 depicts the selection process. Initially, 4530 studies were retrieved from the selected online databases. Then, the duplicates were removed and a few sections of the studies: title, abstract, introduction and conclusion (if necessary to make a decision) were assessed against the inclusion/exclusion criteria. Only 92 out of 4530 initial studies were selected and their full-texts and metadata were recorded in a reference manager named Zotero. Finally, 92 full-texts were read and assessed against quality criteria. Only 60 studies passed the quality threshold and they were selected as final studies. Their references were analyzed and as a result, 10 additional relevant studies were identified. In total, 70 primary studies were reviewed and analyzed in the first part of this PhD project.
- (iii) *Data extraction and synthesis.* An extraction form was created to guide the data extraction process. The extraction form consisted of the following attributes whose data was extracted automatically from Zotero: study title, author(s) name(s), publication date, publication source (journal, conference, workshop, symposium). Additional attributes, such as research methods (survey, experiments, field experiment, case study, solution proposal, or a combination of methods), quality scores, and RT challenges identified and/or addressed were inserted manually by the PhD student. The extraction form and the data can be accessed online in Figshare [126]. According to Kitchenham [122], there are two main data synthesis approaches: descriptive/narrative and quantitative. In this part of the thesis, a descriptive data synthesis approach was followed, meaning that themes were identified and their frequencies were measured. The latter should not be confused with quantitative approaches, since the frequencies of themes facilitate the identification of research gaps and future research directions, rather than suggest their relevance.

3.3.2 Systematic Mapping

The third paper of this PhD thesis (P3) adopted a systematic mapping methodology that relied on Petersen et al. [3]'s guidelines for systematic mappings in SE. Figure 3.3 depicts the systematic mapping methodology which is explained in the following section.

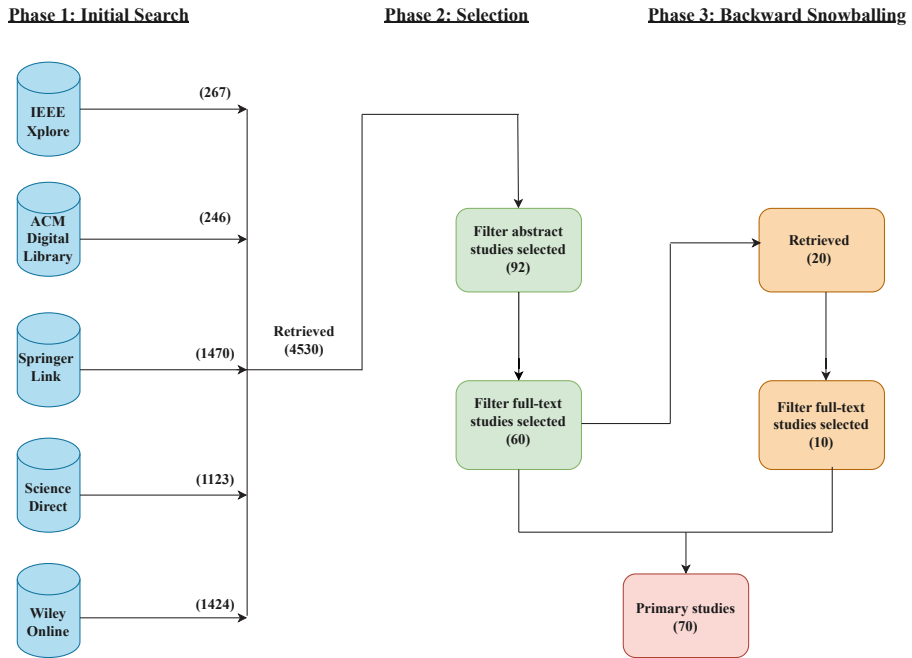


Figure 3.2: Primary studies selection process

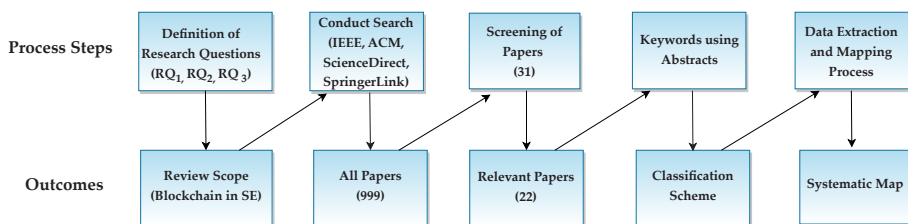


Figure 3.3: Systematic mapping methodology, adapted from [3]

3. Research Methodology

As shown in Figure 3.3, the first step of the process was formulating RQs. In the third paper (P3), four research questions were defined within the main scope of this review study: the uses of blockchain in SE. These questions aimed to provide a comprehensive overview on the trend of studies that use blockchain concepts in SE (RQ1), reported applications (RQ2), blockchain platforms used to develop SE applications (RQ3) and contributions that blockchain brings to the SE discipline (RQ4). The first phase of the search process is the construction of the search string which incorporated relevant terms related to the main concepts of blockchain technology and software engineering, connected via Boolean operators. The search string was executed in 4 databases that are often used in the SE field: IEEE Xplore, ACM Digital Library, Science Direct, and Springer Link, as suggested by [3, 122]. Consequently, 999 studies published up to 2020 were retrieved. The title and abstract of these studies were assessed against a set of inclusion/exclusion criteria and the assessment was passed by 31 studies. In the last selection phase, the full-texts of the studies were assessed against the inclusion/exclusion criteria and a final set of 18 studies passed the assessment. In addition, 4 other studies were selected by means of backward snowballing technique. The final set of 22 studies, the excluded studies and reasons of exclusion can be found in Figshare [127].

The final phase of the systematic mapping process entailed developing the classification scheme and extracting keywords from the selected studies related to three main dimensions: research topic, research type and contribution type, as recommended by Petersen et al. [3]. Clustering these keywords enabled the creation of map categories. In what follows, only the categorization of the three dimensions is presented, but a more detailed explanation of them can be found in the published paper (P3):

- Research topic (software requirements, software engineering process, software testing, software quality, software maintenance, software configuration management, software engineering management, software engineering professional practice)
- Research type (validation research, evaluation research, solution proposal, philosophical study, opinion study, and experience study)
- Contribution type (model, framework, interviews, platform)

The keywords extracted from the studies in each of the aforementioned dimensions can be found in an online repository [127].

3.3.3 Interviews

The method used to improve and validate the proposed framework is experts' judgement by means of semi-structured interviews. Semi-structured interviews were chosen due to their flexible nature and their ability to unveil rich contextual information [128]. Moreover, interviews enable two-way communication between the interviewer (researcher) and interviewee (informant) which leads to a more

personal dialogue and uncovering relevant information [129]. A total of 20 semi-structured interviews were carried out in this PhD thesis with blockchain experts (10; P5) and software engineering experts (10; P7). These interviews were conducted by the PhD student during 2021-2022 through Zoom, recorded and transcribed upon interviewees' consent. The interviews lasted from 40 to 95 minutes. Two different interview guide documents were designed with a set of pre-defined open-ended questions that can be accessed online [130, 131].

In P5, the interviews aimed to uncover blockchain experts' opinions and experience related to the implementation process of blockchain technology in organizational settings. First, the experts were asked about relevant experience with blockchain implementations and the relevant projects they participated. Second, the experts were asked about the process of implementing blockchain technology in organizations, when to implement blockchain technology, how to select the best-fitting platform, challenges and success factors. Although pre-defined questions were formulated to guide the interview process, additional questions were formulated as a result of data analysis. In P7, the interviews aimed to elicit SE experts' opinions on the proposed framework and prototype. Prior to the interview process, the experts were provided with a short description of the framework and with a video in which the PhD student showcased the prototype. Experts were asked about their academic and professional background on software engineering, the usefulness, validity, strengths, and limitations of the proposed framework, along with recommendations on how to further enhance the framework.

3.3.4 Selection of Participants

A total of 20 blockchain or SE experts were selected in this PhD thesis by using purposive sampling technique (P5, P7). In fact, purposive sampling has been considered the most common sampling methodology in SE research [132, 133], and the technique enables researchers to perform expert judgement [134]. In both studies (P5, P7), experts were mainly selected according to their professional and academic experience in blockchain or SE. In the fifth study (P5), Fehring [135]'s experts selection criteria were adapted. It is worthy to note that while it is true that Fehring's criteria were used to select nurses in nursing diagnosis validation studies, these criteria have been also used in the SE field, for instance Herranz et al. [136] adopted the criteria to select SPI experts to validate a gamification-enabled framework for SPI initiatives. The following criteria were used to select blockchain experts:

- Over 2 years of experience (Score=4)
- Academic experience, e.g., teaching and/or supervision of students, related to blockchain (Score=4)
- PhD in blockchain technology (Score=3)
- Blockchain-related articles published in journals (Score=3)

3. Research Methodology

- Blockchain-related book chapters, reports or conference/workshop/symposium papers (Score=2)
- Master’s degree or master’s thesis in blockchain technologies (Score=1)

The candidates were considered experts if they were assessed with a minimum total score of 5. The first part of Table 3.1 shows the main characteristics of the selected blockchain experts (EX_i for $i_{1,10}$) in terms of gender, educational level, years of experience, job position, domains in which they are currently working, and country. It is interesting to outline that all the blockchain experts had more than 2 years of academic or professional experience in blockchain technologies and held different job roles in diverse domains and countries. This diversity provides broader perspectives and brings more information to the analysis. Additional detailed information, including experts’ individual scores is presented in the fifth paper (P5).

A similar strategy was followed in the last validation study (P7) to select SE experts. The second part of Table 3.1 shows the demographics of the selected SE experts (EX_i for $i_{11,20}$). Although the researchers tried to invite a gender-balanced set of SE experts to participate in the validation study, only 3 out of 10 (30%) women agreed to participate. In fact, this reflects the gender-imbalance that still exists in the SE field. Furthermore, the majority of SE experts (90%) had over 10 years of academic and/or professional SE experience, and held a PhD in the SE field. Finally, SE experts held a diverse set of job positions, ranging from SE researcher and professor to software architect, developer and tester in different countries.

Table 3.1: Demographics of blockchain experts and SE experts

ID	Gender	Level of education	Years of experience	Job position	Work domain	Country
EX1	Male	Master	6	Business development director	e-health, energy, supply chain	Norway
EX2	Male	Master	5	Solutions architect	supply chain	US
EX3	Male	Master	4	Blockchain software engineer	all domains	Portugal
EX4	Male	Master	7	Chief executive officer	all domains	Norway
EX5	Male	Master	3	Researcher	marketing industry , education	Sweden
EX6	Male	PhD	6	Associate professor	public sector	Sweden
EX7	Male	Master	8	Chief technology officer	all domains	Spain

EX8	Male	Master	7	Head of blockchain and data strategy	banking	Norway
EX9	Male	PhD	3	Researcher	e-health	Norway
EX10	Male	Master	6	Chief executive officer	all domains	Spain
EX11	Male	PhD	10+	Head of quality and software process improvement	all domains	Austria
EX12	Male	PhD	10+	Software developer, project manager, professor	all domains	Ireland
EX13	Male	PhD	10+	researcher, professor	all domains	Spain
EX14	Female	PhD	10+	researcher, professor	all domains	Mexico
EX15	Male	PhD	10+	Software architect, developer, researcher	all domains	Turkey
EX16	Male	Master	10+	Software engineer	automotive	Egypt
EX17	Male	PhD	10+	Researcher	all domains	Finland
EX18	Female	PhD	10+	Tester, Software Engineer, Product Manager, Professor	all domains	Chile
EX19	Female	PhD	10+	Tester, Professor	all domains	Ecuador
EX20	Male	PhD	5	RE researcher	all domains	Germany

3.4 Data Analysis

3.4.1 Grounded Theory

The roots of grounded theory (GT) can be found in the 1960s in the field of sociology with Glaser and Strauss’s seminal paper “*The discovery of grounded theory: Strategies for qualitative research*” [137]. Despite its origins in social studies, grounded theory has been increasingly employed in Information Systems [138] and SE research [139]. While it is true that software engineering is a technical field, it is also true that human aspects play an important role in designing and developing software, thus the SE community has paid growing attention to such aspects [4, 140, 141]. In this regard, SE researchers have organized the International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE) collocated with the International Conference

3. Research Methodology

on Software Engineering (ICSE) conference since 2011 ¹. It reveals a growing interest in sound qualitative research methods in SE research, such as grounded theory. A quick search in Google Scholar returned several SE studies that adopted grounded theory to (only a subset of studies is mentioned):

- Explore humans aspects of SE [139]
- Explore practices of self-organizing Agile teams [142]
- Investigate how much upfront architecture design effort is enough according to agile software teams [143]
- Develop an Agile transition and adoption framework [144].

Grounded theory is fundamentally a method of generating theory from inductive analysis of empirical data [137]. Despite the existence of different GT variants (the three main variants are explained in the following section), there are a set of core components that all the variants share, albeit implement them differently:

- *Delay the scrutiny of extant literature.* Often, the GT researcher is expected to start the data collection process without first conducting a literature review. This has been referred to as “the blank slate” of the GT researcher [138]. However, according to Urquhart and Fernández [138], this is a misconception or misinterpretation of the GT principle of “setting aside the extant theory”. While this principle does not imply that the literature should be ignored, it does emphasize the importance of detachment to avoid possible biases and preconceptions that might influence the emergent categories and theory. In fact, the literature review has been recommended to be used in two phases [138]:
 - (i) non-committal phase. This phase consists of a pre-study review to define the problem area and understand the methodology. This helps the researcher to develop theoretical sensitivity.
 - (ii) integrative phase. The researcher is recommended to return to relevant literature and integrate it with emergent findings in two stages [138]. The first stage is once theoretical concepts emerge in order to compare them with converging and diverging patterns in literature. These comparisons can stimulate new ideas and lead to further theoretical sampling to reach saturation. The second stage is once the core category and theory emerge, because it allows positioning the emergent theory within the wider literature to enhance its robustness and value.
- *Simultaneous data collection and analysis.* Differently from other qualitative methods, the GT researcher does not have to wait until the data collection process finishes in order to start analysing the data. Instead,

¹CHASE was considered a workshop from 2011 to 2020 and a conference from 2021 to 2023. For more information, refer to <https://www.chaseresearch.org/workshops>.

collected data is analysed prior to subsequent data collection. The analysis helps to identify gaps and unsaturated concepts which guide subsequent data collection phases.

- *Theoretical sampling.* The identified gaps and the need for additional saturation of concepts lead to the identification of additional data sources.
- *Theoretical sensitivity.* Theoretical sensitivity is the ability of GT researchers to develop concepts and identify relationships between concepts. While creativity is important in this process [145], the integration with relevant literature can be of value [138].
- *Theoretical saturation.* The theoretical saturation point is the point at which there is no need to collect and analyse additional data because the theory is already well-grounded in empirical data and new data does not trigger any updates of the emergent theory.
- *Memoing.* The GT researcher writes memos in the form of notes or diagrams to describe in detail categories, their properties and relationships between categories.
- *Constant comparison.* Grounded theory relies on continuously comparing data, codes, categories and memos. For instance, codes identified in an interview transcript are compared with other codes in that transcript and previous transcripts, meaning that codes and categories are always evolving.
- *Theoretical sorting.* Theoretical sorting is a continuous process of moving back and forth between memos and the theory to position properly the categories that were created as a result of the coding process.

It is worthy to note that the list of components is not exhaustive. For information regarding additional components, the interested reader is referred to Glaser, Strauss and Corbin's books [137, 145].

There are three main GT genres that evolved from the original GT discovered by Glaser and Strauss in 1967 [137]:

- (i) *classic or traditional GT* associated with Glaser [146]. Traditional GT is rooted in the philosophical paradigm of objectivism meaning that there is a single, accurate description of reality, therefore concepts and theory can be discovered from empirical data.
- (ii) *evolved GT* associated with Strauss and Corbin [146]. Evolved GT is rooted in symbolic interactionism which goes beyond objectivism and takes a more interpretivist approach [139]. According to Chun Tie et al. [147], symbolic interactionism focuses on subjective meaning people give to events, objects and behaviours. In such a case, reality is created through interactions which in turn are based on language and communication [139].

3. Research Methodology

- (iii) *constructivist GT* associated with Charmaz [148]. Constructivist GT is rooted in constructivism in which the constructivist (researcher) and participants co-construct meanings and experience. This collective action forms the social reality in which the researcher is not neutral.

Despite the philosophical foundational differences between GT genres, there are other differences related to the formulation of RQ, the role of literature and coding techniques. The reader is referred to Stol et al. [139]’s comprehensive comparison of the three GT genres.

Traditional GT was adopted in the fifth study of this PhD project (P5) to explore the implementation of blockchain technology in organizations. The underlying motivations behind this choice are presented, as follows:

- Blockchain technology has been conceptualized as a socio-technical assemblage [149]. Hence, explorative research on blockchain technology can leverage GT, given that GT is rooted in social sciences and it is suitable to explore phenomena with social implications.
- GT is used to explore novel, multi-dimensional phenomena [150]. While blockchain has been increasingly used in several domains, blockchain research is not yet mature from a theoretical, empirical and methodological point of view [151].
- GT is suitable for inductive studies, rather than deductive studies that aim to test upfront hypothesis. The study (P5) adopted the GT methodology to generate categories grounded in empirical data which were used to enhance the blockchain-enabled RT framework proposed in the fourth study (P4).

Figure 3.4 depicts the GT process followed in the study (P5), adapted from Hoda et al. [4].

Interviews generate rich qualitative data that if managed manually, may lead to inaccurate data analysis. To minimize the errors and facilitate the process, Nvivo software was used. This software has been previously used in SE studies that use GT, e.g., [144]. The data analysis process in the fifth paper (P5) adopted the following traditional GT techniques, as recommended by Glaser [152]:

- (i) *open coding*. First, the PhD student read the transcripts to have an understanding of the context under study. Second, each transcript was reread with the goal to identify key points. Third, labels referred to as codes were assigned to each key point. Finally, the codes were compared with other codes in the same transcript and previous transcripts. This constant comparison process is important in GT, as it facilitates the discovery of concepts and categories.
- (ii) *core category*. The discovery of the core category terminates the open coding process. The core category must encompass the main concerns raised by interviews. Moreover, the GT researcher should consider three main criteria when choosing the core category: (i) Is the core category

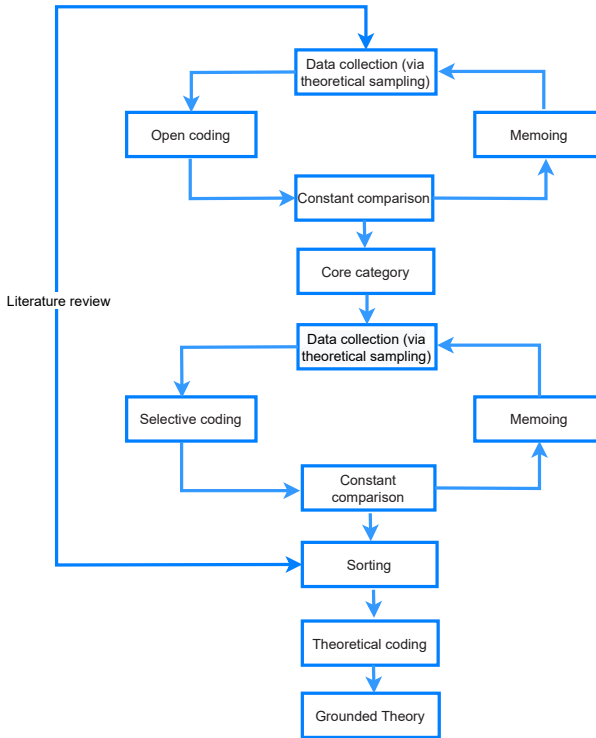


Figure 3.4: Grounded Theory phases, adapted from Hoda et al. [4]

central? (ii) Is the core category related to the other categories in a meaningful manner? (iii) Does the core category account for the majority of data variations?

The process of identifying the core category is not trivial, but it can be facilitated with the use of the constant comparison technique on categories and the identification of relationships between categories. In this PhD study (P5), the first 5 interviews pointed out feasibility analysis as a candidate core category, because it was related to the rest of the categories meaningfully. However, a further comparison process indicated that this category was not central and revealed blockchain implementation in organizations, as the most central category. The identification of the core category is followed by selective coding which focuses on coding only the core category and associated categories.

(iii) *theoretical memoing*. This process consists of writing memos in Nvivo

3. Research Methodology

regarding perceptions, thoughts or reflections on emergent categories. These memos are useful in identifying relationships between codes.

- (iv) *sorting*. The sorting process starts after the data collection process and focuses on explaining categories and their relationships. Given the limited generalizability due to the relatively low number of interviews, a minimal literature review was incorporated to the findings in this phase of the process (P5).
- (v) *theoretical coding*. Theoretical coding focuses on core categories-associated categories relationships, contributing in this way to the formulation of the emergent theory. Theoretical coding families proposed by Glaser [152] can be useful to explain relationships and the emergent theory. The fifth study (P5) is based on the temporal/process coding family to represent the category key activities in terms of timeliness, stages and temporal work ordering [144], as depicted in Figure 3.5.

Finally, Figure 3.6 represents a sample of the coding process that demonstrates the emergence of the category *challenges* from a set of identified codes and concepts.

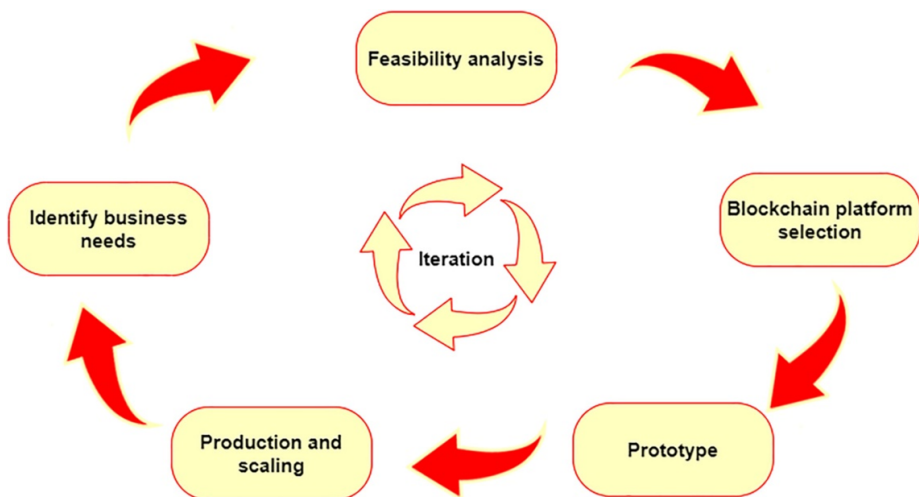


Figure 3.5: Representation of *key activities* category by means of Glaser's temporal/process family

3.4.2 Content Analysis

While the roots of content analysis can be traced back to the beginning of the use and analysis of symbols and texts in ancient fields, such as philosophy,

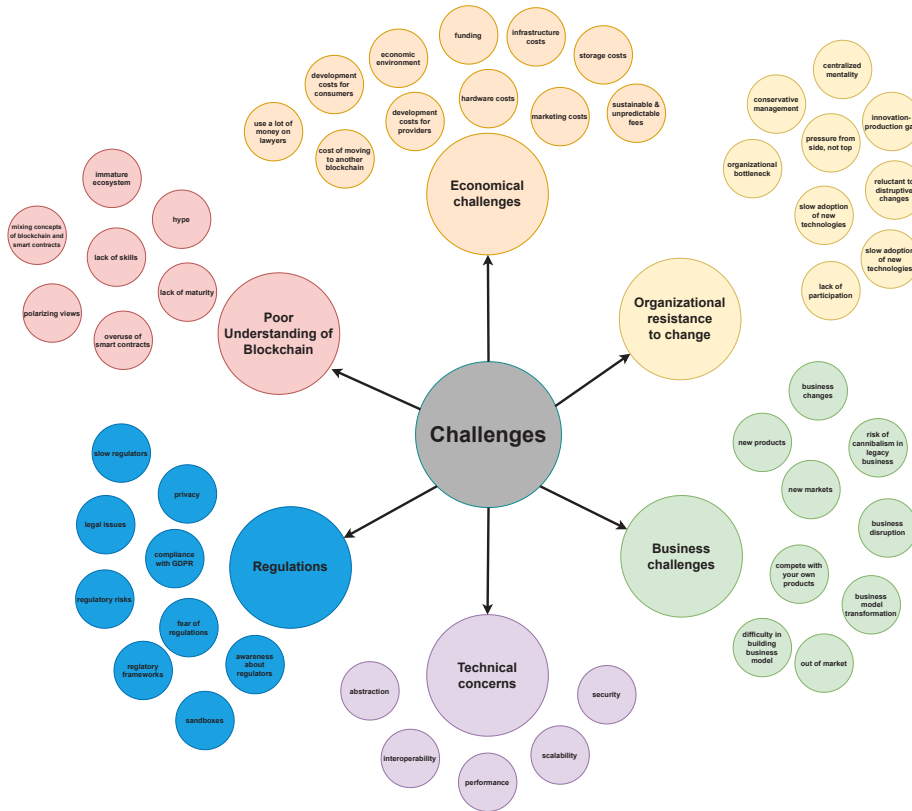


Figure 3.6: Mind map to depict the emergence of the category "Challenges" from underlying concepts and codes

today's content analysis is a method with exploratory nature that is grounded in empirical data [153]. During the last two decades, content analysis has been adopted in SE studies, for instance to identify communication issues in requirements elicitation through stakeholders' lenses [154], to explore the main dimensions of SE success [155], and to explore the rationale of users in software reviews [156]. Hsieh and Shannon [157] presented the three following content analysis approaches and the differences among them:

- (i) *conventional content analysis*. In conventional content analysis, the researcher generates coding categories from data.
- (ii) *directed content analysis*. In directed content analysis, existing theories or previous research findings are used to guide initial codes which are then used to analyse empirical data.
- (iii) *summative content analysis*. In summative content analysis, the researcher counts and compares keywords or content, and interprets the underlying

3. Research Methodology

context. These keywords can be defined before starting the data analysis process using a literature review.

Content analysis was chosen as the main analysis method in the last paper of this PhD project (P7), because of its flexible design nature [158], and the fact that it puts emphasis on the content of words [153]. The last study (P7) used the content analysis process introduced by Elo and Kyngäs [5], due to the high number of citations (2960 citations related to SE out of 22389 total citations, Google Scholar, accessed on 03.12.2022). The process is depicted in Figure 3.7, and consisted of three main phases:

- (i) *Preparing.* The first phase starts with identifying the unit of analysis which can be a word, theme, one or more sentences, entire interviews or the number of subjects. If the unit of analysis is too broad, the analysis process becomes challenging, but if the unit analysis is too narrow (e.g., one word), fragmentation can occur [5]. In P7, entire interviews are chosen as the unit of analysis, in line with Graneheim and Lundman [159]’s recommendation that considered interviews as large enough to obtain a sense of the whole, and small enough to contextualize meaning units during the data analysis process. Then, the interviews’ transcripts were uploaded in Nvivo and were read by the authors of the study to acquire a sense of the whole.
- (ii) *Organizing.* After understanding the data, the analysis was carried out using an inductive approach. Following the inductive approach, the PhD student organized the qualitative data by means of open coding, content-based grouping, categorization and abstraction. The transcripts were broken down into one or more sentences that represented meaning units which were transformed into condensed meaning units by reducing and simplifying the wording. Then, codes were assigned to meaning units in an iterative fashion, were collected in coding sheets and were grouped based on their content into categories. To enable a higher abstraction level, categories can be further categorized until a logical and understandable explanation has been reached.
- (iii) *Reporting.* Finally, contents of the categories were described in detail. This phase is necessary to demonstrate links between the results and empirical data, thus ensure reliability and transparency of the study [160, 161]. Samples of the coding process can be found in Table 3.2 and in an online repository [131].

3.5 Reflections on Methodological Quality

In this section, the credibility of the methodological decisions made in this thesis is discussed in terms of validity, generalizability, and ethical considerations.

Table 3.2: Sample of the coding analysis process

Meaning unit	Condensed meaning unit	Code	Sub-category	Category
<p><i>“It could be too much details like most of the time as you’re working in the same field, this kind of high complex documentation is one of the things you need to avoid at some stage. So, you are creating enormous amounts of documentation or just processing work...”</i></p>	<p>Complex and enormous documentation is being created. High amount of processing work (overhead)</p>	<p>Documentation perceived as overhead</p>	<p>Business</p>	<p>Blockchain-based software process improvement</p>
<p><i>“In my experience, there are software engineers..., who are used to use technologies which they are familiar with, and they are quite reluctant to embark on new technologies. That’s a little bit contradictory to their capacity to change, because they would understand the technology.”</i></p>	<p>Software engineers are reluctant to embark on new technologies</p>	<p>Software engineers’ resistance to change</p>	<p>Change</p>	<p>Blockchain-based software process improvement</p>
<p><i>“One thing that is missing is how to evaluate the return on investment of this technology. I think it is missing a sixth step «Evaluate», because you have the prototype, but it is important to know how much it is going to cost”</i></p>	<p>Missing final step: Evaluate return on investment</p>	<p>Add a final phase ‘Evaluate’</p>	<p>Experts’ recommendations</p>	<p>Experts’ recommendations</p>

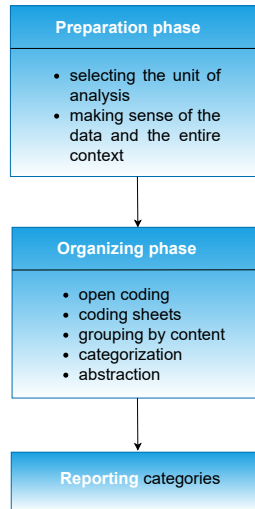


Figure 3.7: Content analysis process, adapted from Elo and Kyngäs [5]

3.5.1 Validity

According to Cook and Campbell [162], validity refers to “*the best available approximation to the truth or falsity of propositions*”. An interpretation of this definition can be that research per se is not subject to validity, rather the conclusions drawn from the results of the empirical study may be subject to validity [163]. Indeed, studies’ design choices and their execution influence the validity of the results and conclusions. Validity can be discussed in terms of internal and external validity. This section focuses on the former, while the latter is explained in Section 3.5.2. Internal validity can be defined as the extent to which the researcher is confident in conclusions drawn from a cause-effect relationship [162]. The validity of results and conclusions can be threatened by researchers’ biases. The presence of researchers’ biases is acknowledged in this thesis, particularly in the following stages of review studies (P2, P3): in the selection of databases, in the formulation of search strings, inclusion/exclusion criteria, design of extraction forms and interpretation of findings. To minimize these biases and to ensure accurate description of findings that lead to conclusions grounded in data, these processes were first performed by the PhD student based on a protocol and then reviewed by other researchers with experience in conducting SE secondary studies.

In addition to systematic reviews, threats to internal validity are present in qualitative research, for instance possible biases in the selection of blockchain and SE experts (P5, P7). To address this threat, a strategy was defined and used to select experts with relevant academic and professional experience in

blockchain and SE, in a systematic manner (See Section 3.3.4). Following such a systematic approach is very important, as the selection of experts affects the validity of results and conclusions. Another threat to internal validity worthy to be mentioned is the abundance of data generated in qualitative research which leads to a time-consuming, tiring and error-prone coding process, in particular for novice researchers [164]. To address this limitation, the PhD student was supported in her inquiry journey by two experienced researchers in the SE field, as recommended by Annells [165] and Chun Tie et al. [147].

It is also worthy to outline that this thesis follows a multi-methods approach, as explained in Section 3.2. This approach enables methods triangulation which is one of the main strategies used to enhance internal validity of qualitative approaches [166]. Finally, publication bias can be a potential threat to validity, due to the nascent phase of the blockchain-enabled SE field, in particular the lack of studies in the blockchain-enabled RT topic. However, this is perceived as supporting the novelty of the thesis, hence facilitating the publication of studies, rather than hindering the publication process.

3.5.2 Generalizability

Although the findings of this thesis are limited to the scope of the research project, further research can contribute to transferring the findings across a variety of domains and contexts. For instance, the thesis focused mainly on tracing requirements, source code artifacts, test cases and test results in general-purpose domains, but further research can extend the proposal by tracing other artifacts in safety-critical systems, e.g., in the automotive domain. The framework can also be adapted to specific software development practices, such as agile practices, and include other stakeholders' roles and artifacts. In fact, the findings of all the studies in this thesis were intended to be domain-agnostic, for instance challenges of RT were identified independently from the domain (P2). In addition, blockchain experts from different domains, countries and blockchain-related roles were interviewed to unveil the implementation process of blockchain in organizational settings (P5). This was done on purpose to gain a broad perspective on the topic of interest. It is also worthy to mention that achieving generalizability in qualitative research is complex, due to the focus on the contextualized understanding of human opinions and experience [167]. Achieving generalizability is particularly challenging, when the number of interviews is low [168]. Given the lack of unanimously agreed guidelines about sample size in qualitative research [154], Morse [169]'s factors were adopted in P7 to determine the number of SE experts:

- (i) *focused scope of study*. The scope of P7 was to validate blockchain-enabled RT framework and prototype.
- (ii) *clear nature of the topic*. The topic investigated in P7 was trustworthy traceability management by means of blockchain technology.

3. Research Methodology

- (iii) *data quality*. Rich empirical data was collected in over 100 pages of interview transcripts in the last paper (P7) and a set of 84 codes emerged.

While these factors cannot ensure conclusiveness of findings, they suggest that the number of experts satisfies the purpose. Moreover, it is worthy to note that the goals of P5 and P7 lie in refining the proposed framework and validating the proposal respectively, rather than achieving generalizability of findings. The GT approach adopted in P5, like any other qualitative inquiry method, is inevitably prone to external validity threats [167], although the 4 criteria: fit, workability, relevance and modifiability proposed by Glaser [152] were used to validate the categories that emerged in P5. After the first 8 interviews conducted with blockchain experts, it was observed that new categories did not emerge, suggesting the theoretical saturation point. Although two more interviews were conducted to confirm the theoretical saturation point, it is worthy to highlight that the theoretical saturation point does not ensure generalizability.

Finally, qualitative research should ensure reproducibility which refers to the ability to yield the same results, by adopting similar raw materials [170]. To enable reproducibility and transparency, the analysis processes followed in this thesis (P2, P3, P5, P7) are documented in Figshare, as transparent reproducibility packages [126, 127, 130, 131].

3.5.3 Ethical Considerations

This PhD dissertation contains two studies that carried out interviews with 20 blockchain and software engineering experts. Although the studies did not handle sensitive data, they were submitted for approval to the Norwegian Center for Research Data (NSD), prior to their execution. It is worthy to note that informed consent forms were created for each of the studies and they are available online in Figshare [130, 131]. These forms consisted of detailed information regarding the rights of participants, such as voluntariness and anonymity, possible implications of participation in these studies, the purpose of the studies, motivations behind the selection of experts, information regarding data recording, storage and deletion, along with benefits and risks of the studies. The informed consent forms were evaluated to ensure adherence to General Data Protection Regulation (GDPR) and were approved by the Data Protection Officer at Østfold University College.

Chapter 4

Research Findings

“If we knew what we were doing, it would not be called research, would it?”

(Albert Einstein)

This section provides an overview of 7 papers that comprise the underlying basis of the thesis. The papers have been published in international conferences or Scopus-indexed journals that are ranked in Journal Citation Report. The papers were designed based on the initial PhD research project that received feedback in the RE Doctoral Symposium. Diverse data collection and analysis approaches were adopted in the papers to address the research questions (See Section 3.2). Reflections regarding the purpose, research approach, findings and contributions of the papers are presented briefly in this section, in the order in which the papers were written during the PhD program.

4.1 Paper I: Blockchain-oriented Requirements Engineering: A Framework

S. Demi *"Blockchain-oriented Requirements Engineering: A Framework,"* in 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 428–433, IEEE, 2020.

Overview: The first paper [171] provides an overview of the initial PhD research plan. The paper emphasized the motivation for starting the PhD project, related work in the fields of requirements traceability process, technologies and tools, requirements traceability challenges, and the bi-directional relationship between blockchain technology and software engineering. In addition, the paper outlines the purpose of the PhD research project, the planned research process and activities, data collection methods and analysis techniques. The paper was submitted at a Doctoral Symposium which was part of the well-recognized International Requirements Engineering Conference 2020. Potential benefits and risks of the proposed approach were discussed with RE experts. A mentor was assigned to each PhD student to provide feedback on the PhD plan. This process contributed significantly to improving the initial plan and guiding the work of the PhD student in a systematic manner.

4.2 Paper II: What have we learnt from the challenges of (semi-) structured requirements traceability?

S. Demi, M. Sanchez-Gordon, and R. Colomo-Palacios, "What have we learnt from the challenges of (semi-) automated requirements traceability? A discussion on blockchain applicability," IET Software, vol. 15, no. 6, pp. 391–411, 2021.

Purpose: The RT community has investigated RT challenges for over three decades. The core work in the field has been conducted by Gotel and Finkelstein [11], followed by CoEST [13] that identified eight RT challenges: purposed, cost-effective, configurable, portable, trusted, scalable, valued and ubiquitous. The second study of this PhD project [103] aimed to contribute to the existing knowledge in the field by identifying and categorizing challenges of (semi-)automated RT, as reported by empirical studies published during the period 2009-2019. The need for a holistic and updated view on the challenges was identified as a result of a preliminary literature review that suggested the lack of such studies in recent literature. In fact, the last similar study has been carried out in 2009 by Winkler and Pilgrim [31].

Research approach: This study followed a SLR approach based on Kitchenham and Charters's guidelines for performing systematic reviews in software engineering [123]. The research approach consisted of the following phases:

- (i) *Planning the review.* First, the main research question was formulated relying on PICO (population, intervention, comparison, outcome) strategy, then the search strategy was defined in terms of search string and online databases. It is not surprising that database searches produce a high number of studies that often are irrelevant to the research topic. Therefore, the retrieved studies should go through a rigorous selection process. This process started with defining inclusion and exclusion criteria.
- (ii) *Conducting the review.* The selection of studies went through three different phases. First, the search string was executed in four major databases and a set of 4530 studies was retrieved. Second, the title, abstract, introduction and conclusion of the studies were evaluated using the inclusion and exclusion criteria. The second phase produced 92 studies and their full-texts were recorded in Zotero. Finally, the authors of the study read the full-texts and selected the final set of studies based on pre-determined quality criteria. As a result of this phase, 60 studies were selected and 32 studies were excluded. Snowballing was performed on the final studies to identify additional primary studies that might had been missed throughout the process. Consequently, 20 studies were discovered, and only 10 of them passed the assessment. In total, the review encompassed a set of 70 relevant studies.
- (iii) *Data extraction and synthesis.* An excel extraction form was used to collect data for each study. The following data were collected: title, author(s)

name(s), publication date, publication source, research methods and themes with respect to RT challenges. Descriptive data synthesis was followed by identifying and measuring themes from the extracted data.

Findings: The main findings of this SLR are presented in the following section:

- The majority of RT primary studies were published in conference proceedings. In fact, 32% of the studies were published in the proceedings of the International Requirements Engineering conference which is the most important venue in the field. This means that a significant part of the traceability research has been performed by the RE community, in line with the findings of previous studies [31].
- Over 50% of the primary studies adopted experiments as the main research method, and a lack of exploratory studies, such as case studies and surveys, was observed. This finding suggested the need for more exploratory studies to enhance the understanding of traceability practices in organizational settings.
- The analysis of the primary studies revealed two main directions of traceability research with respect to challenges: (i) approaches focused on solving (semi-)automated traceability generation and maintenance challenges, (ii) approaches focused on challenges related to the human facet of traceability. In addition, the review indicated the low accuracy of RT recovery methods, as the most frequently addressed challenge.
- The study identified a set of 21 challenges which were then grouped into 5 main categories, as follows: (i) technological challenges (low accuracy of traceability recovery methods, inadequate integration or interoperability among heterogeneous tools, traceability decay, lack of change notification and propagation, poor presentation and visualization of trace links), (ii) human factors (lack of trust in human's judgement, lack of system experience, lack of training, invisible benefits, provider-user gap, perceived as overhead), (iii) organizational challenges (lack of organizational strategies and guidelines for traceability, undefined roles and responsibilities for traceability, project dimensions related challenges, challenges enabled by the software development approach), (iv) communication and collaboration challenges (intraorganizational communication challenges, communication challenges in distributed software development, interorganizational collaboration challenges), and (v) regulatory challenges (implicit traceability requirements in regulations, granularity in requirements for traceability, legal and intellectual property constraints).
- Recent studies advocated for new technologies to solve RT issues [20]. In the same line, this study discussed and elaborated on the potential of blockchain to enable trustworthy, reliable and available RT across organizational boundaries.

Contributions: The contributions of this paper can be categorized into three main dimensions: (i) providing a holistic and domain-agnostic view of RT challenges by identifying 21 challenges and grouping them into 5 categories, (ii) discussing the novel application of blockchain technology for ensuring trustworthy and available traceability in interorganizational software projects, and (iii) unveiling research gaps that require future research efforts, such as distributed traceability, traceability approaches in agile and DevOps, investigation of human factors in RT, and the need for more exploratory studies to increase the understanding of traceability practices in organizational environments.

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

S. Demi, R. Colomo-Palacios, and M. Sanchez-Gordon, *Software Engineering Applications Enabled by Blockchain Technology: A Systematic Mapping Study*, Applied sciences, vol. 11, no. 7, p. 2960, 2021.

Purpose: The potential benefits of blockchain technology have been investigated in a variety of domains [106, 107, 172]. Recently, SE researchers have explored the bi-directional relationship between blockchain technology and SE [108, 109]. The third study of this thesis [110] aimed to present a holistic, quantitative and qualitative overview of applications of blockchain technology in SE. The main aim was decomposed into the following four deliverables:

- an overview of the trend of studies that use blockchain in SE and their research methods,
- identification of blockchain use cases and their classification into SE knowledge areas. This classification was important to outline overlooked areas.
- an overview of implementation aspects of blockchain-enabled SE applications, particularly blockchain platforms and consensus mechanisms.
- identification of the contribution of blockchain in the SE domain, by mapping blockchain properties with respective SE challenges.

Research approach: This study adopted a systematic mapping approach that followed Petersen et al. [3]’s guidelines for systematic mappings in SE. The following search string was developed considering the main scope of the review which is “blockchain in SE”: (*blockchain OR “smart contracts” OR “distributed ledger”*) AND (*“software development” OR “software engineering”*). This search string was executed in 4 major databases: IEEE Digital Library, ACM digital library, Science Direct, and Springer Link. This process returned 999 initial studies which were then assessed against inclusion/exclusion criteria. A set of 18 studies were evaluated as relevant. Then, the references of these studies were

scanned and 4 additional studies were discovered. A total of 22 studies composed the final list of relevant studies.

The last phase of the process entailed the development of the classification scheme by using the keywording technique. Keywords were extracted from the abstracts of the selected studies and then, they were grouped to create map categories. Map categories were created in three facets:

- *research topic*. This facet aimed to structure the topics related to blockchain use cases in SE knowledge areas.
- *research type*. According to Petersen et al. [173], the main research types are evaluation research, validation research, solution proposal, philosophical, opinion and experience study.
- *contribution type*. The main contribution types of this study: model, framework, interviews and platform were adapted from Petersen et al. [173].

Findings: The main findings of this study are summarized, as follows:

- The first study was published in 2015 and 91% of the studies were published during 2018-2020 which suggests slightly growing efforts in the field. All the studies were published in workshops and conference proceedings, consequently a lack of journal studies was discovered.
- 95% of the studies proposed solutions or validated their proposals in experimental settings. In addition, 77% of the studies contributed with models or frameworks. The identification of only one evaluation study indicates scarce empirical evidence regarding the impact of blockchain technology in SE.
- The study depicted findings in a scatter plot that presented the intersection of the research type and contribution type dimensions with the research topic dimension (See Figure 4.1). The identified use cases were grouped into 8 SE knowledge areas, adapted from SWEBOK: software requirements, SE process, software testing, software quality, software maintenance, software configuration management, software engineering management, and professional practice. A more detailed description of the use cases can be found in P3.
- The most used blockchain platforms are Ethereum and Hyperledger Fabric, due to their inherent ability to execute code. Only two consensus mechanisms were proposed in the selected studies:
 - a consensus mechanism that is based on a probability distribution function which takes as input developers' activity in terms of their contribution and waiting time, and number of votes of testers for each developers' work, i.e., code.

4. Research Findings

- proof-of-skill which relies on maintaining skill rating for testers based on their previous performance. Testers are grouped based on their votes and the average skill rating is computed for each group. The group with the highest skill rating wins the dispute.
- Blockchain properties, e.g., decentralization, transparency and trust, immutability and data security, anonymity, non-repudiation, smart contracts, have been mapped with SE challenges. To summarize, blockchain can contribute to the SE domain in 4 main directions:
 - Use decentralized systems based on blockchain to ensure availability and eliminate single point of failure and compromise. These systems can replace existing centralized systems, such as GitHub, Travis CI, and cloud-based package managers.
 - Blockchain can function as backbone of the SDLC ecosystem where all software artifacts and their properties can be stored and shared among stakeholders in distributed locations. By doing so, blockchain can enhance trust among stakeholders and enable auditability, compliance to standards and regulations analysis, and identity assessment analysis.
 - Reliable software artifacts sharing by detecting illegitimate accesses or modifications of software artifacts.
 - Smart contracts can automate a set of SE activities that rely currently on humans, for instance, verification of acceptance criteria, verification of quality criteria, compliance to standards and regulations, and automatic payments to developers.

Contributions: The blockchain-oriented SE field is not yet mature from a research perspective, as this secondary study is one of the first attempts to organize and advance the existing knowledge in the field. The findings of this study can be valuable to both academia and industry, particularly the findings may benefit the following stakeholders: (i) researchers interested in exploring the potential of blockchain technology in the SE landscape, and (ii) practitioners willing to comprehend how this novel technology can disrupt the software development industry. Researchers can take inspiration from the use cases presented in this review and contribute with prototypes and proof-of-concepts to enhance the understanding of blockchain technology applications in SE. Strong empirical evidence on the benefits of using blockchain to address SE issues can motivate SE practitioners to implement the proposed blockchain-enabled applications in their organizational settings.

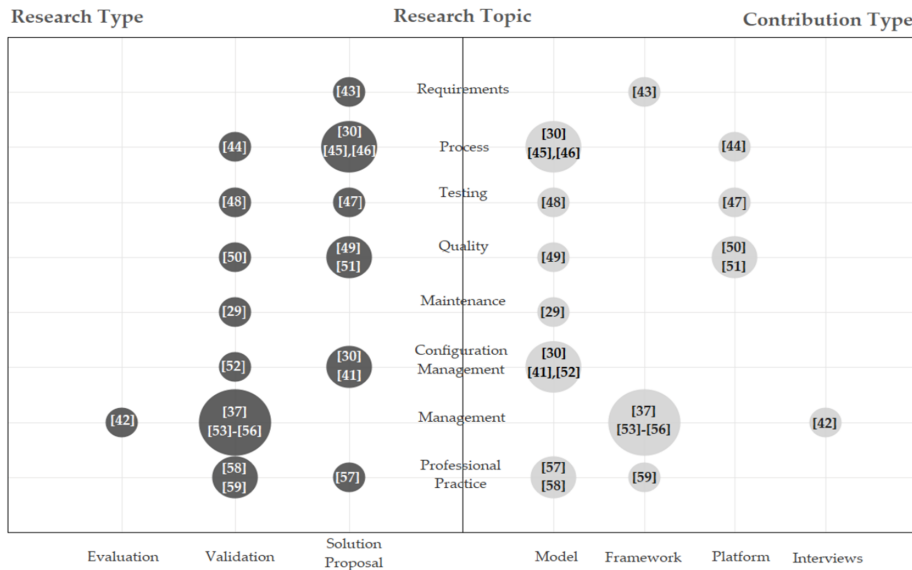


Figure 4.1: Systematic map visualization using bubble plot

4.4 Paper IV: A Blockchain-enabled Framework for Requirements Traceability

S. Demi, M. Sánchez-Gordón, and R. Colomo-Palacios, "A Blockchain-Enabled Framework for Requirements Traceability," in Systems, Software and Services Process Improvement: 28th European Conference, EuroSPI 2021, Krams, Austria, September 1–3, 2021, Proceedings, pp. 3–13, Springer, 2021.

Purpose: Enabling traceability is particularly challenging in projects with distributed teams. Existing centralized artifacts repository can be subject to internal and/or external attacks performed by entities with malicious intentions. In addition, many projects require the participation of third-party vendors, and organizational boundaries in such a case inevitably exacerbate trust issues. The fourth study of this PhD project [6] proposed a blockchain-based framework for trustworthy RT with the ultimate goal of enabling a reliable traceability knowledge base that keeps track of source and destination artifacts throughout the software development lifecycle. The objectives of the framework are as follows:

- Provide stakeholders with a holistic and reliable view of the software development lifecycle.
- Incentivize software engineers to participate in traceability tasks.

4. Research Findings

- Use voting mechanisms to validate the authenticity and quality of trace links.
- Use query services to enhance the understanding of traceability information.
- Trace links should be visualized in a hierarchical and interactive manner.

Solution Proposal: The initial proposed framework is depicted in Figure 4.2. In what follows, the main components of the framework are explained:

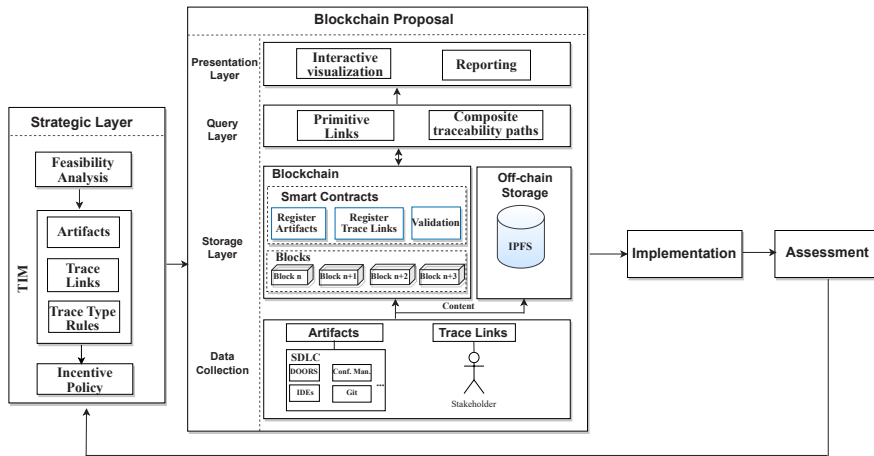


Figure 4.2: Initial blockchain-oriented RT framework, published in [6]

- Strategic Layer.* This layer encompasses the strategic decisions that need to be made by software organizations interested in implementing blockchain technology. The paramount decision is whether blockchain technology is needed and feasible in the specific organizational environment. To facilitate the decision-making process, organizations should conduct feasibility analysis that consists of alignment analysis between blockchain properties and RT strategies, and technological comparison analysis between blockchain and existing tools and technologies used for storing traceability information, e.g., traceability matrices in Excel, relational databases, graph traceability repositories, and cost-benefit analysis. Then, the organization should define a traceability information model (TIM) which can be composed of the following elements: artifacts to be traced and their metadata, relations between source and destination artifacts, and trace type rules that refer to trace links' semantics. These elements can be encoded into smart contracts to allow only the registration of the artifacts that are specified in TIM and identify trace links' naming automatically.

The organization should also define incentive policies that consider the eligibility to register trace links, validation of quality trace links and the amount of incentives that should be assigned for the creation of each trace link type, according to its priority.

(ii) *Blockchain Proposal.* The architecture of the proposal entails four main components, as follows:

- *Data Collection.* Artifacts specified in TIM should be captured automatically by means of data ingestion tools/plugins and then recorded on blockchain. SDLC entities create trace links manually or by means of (semi-)automated RT tools and invoke respective smart contracts to register trace links.
- *Storage Layer and Smart Contracts.* A set of functions are enabled by smart contracts: register artifacts, register trace links, validate trace links authenticity and quality, and reward creators of quality trace links. Artifacts' contents can be stored off-chain in IPFS (Interplanetary File System) to address storage limitations.
- *Query Layer.* Traceability queries can be composed for primitive links, i.e., between adjacent artifacts, and composite trace links, i.e., between non-adjacent artifacts.
- *Presentation Layer.* The framework suggested hierarchical and interactive representation of trace links.

(iii) *Implementation.* A system based on the framework should be developed by establishing close collaboration between software engineers and blockchain experts or startups. Other important aspects that should be considered in the implementation phase is defining metrics for the selection of the best-fitting blockchain platform and communicating the value of using blockchain for RT clearly to all SDLC entities.

(iv) *Assessment.* The contributions of blockchain in the software development lifecycle should be assessed. Potential contributions can be:

- Motivate SDLC entities to create quality trace links.
- The holistic view of trace links and their visualization may encourage the use of traceability in supporting other SDLC activities.
- The increased use of traceability can enhance the performance of software engineers in addressing SDLC tasks.
- Enhanced performance leads to software quality, which in turn results in less need for software maintenance and consequently, cost savings.

Contributions: The findings of the literature reviews conducted in the initial phase of the PhD project (P2, P3) suggested that this is the first framework that uses blockchain for reliable requirements traceability. This initial framework provided the basis for the next studies of this thesis, which aimed to improve it,

particularly from a feasibility perspective. The framework was first improved according to the judgement of blockchain experts and then evaluated by SE experts. This initial framework can also encourage other researchers to build prototypes based on blockchain platforms, such as Ethereum or Hyperledger Fabric to validate the concept.

4.5 Paper V: Blockchain for Requirements Traceability: A Qualitative Approach

S. Demi, M. Sánchez-Gordón, and M. Kristiansen, "*Blockchain for Requirements Traceability: A Qualitative Approach*," *Journal of Software: Evolution and Process*, p. e2493, 2022.

Purpose: Blockchain has received significant attention in industry and academia. The preliminary research conducted by the PhD student indicated that most of the research takes a technical, rather than an organizational perspective. Therefore, it is important to provide evidence on how blockchain technology is implemented in organizational settings. The fifth study of this thesis [174] aims to shed light on the implementation process of blockchain technology from the lens of blockchain experts. The findings are used to improve a previously proposed blockchain framework [6]. The ultimate goal of this improvement is to facilitate the implementation of the proposed framework in software-oriented organizations based on practical recommendations.

Research approach: This study followed the traditional GT approach which aims to generate theories or frameworks based on inductive data analysis [137, 147]. The motivations behind the choice of this research method are explained in Chapter 3. The GT approach is characterized by an ongoing and iterative relationship between data collection and data analysis. This means that the concepts derived from the analysis of the initial interviews' data are used to formulate focused questions in the next interviews. The data collection method used in this study is semi-structured interviews with blockchain experts. Semi-structured interviews were chosen, because they enable flexible discussions between the interviewer and interviewee [128]. A more detailed description of the experts' selection process and interviews is presented in Chapter 3.

Findings: Data analysis suggested the emergence of the core category: "*Blockchain implementation in organizations*" and three related categories: *key activities, challenges, and success factors* (See Figure 4.3). The main findings are summarized in the following section:

- Business needs and requirements should drive the choice for blockchain technology and not vice versa. Blockchain is not suitable for every use case.
- According to our experts, the identification of the right environment takes 60% of the entire process of implementing blockchain technology. In fact, the importance of a critical and thorough investigation about the need

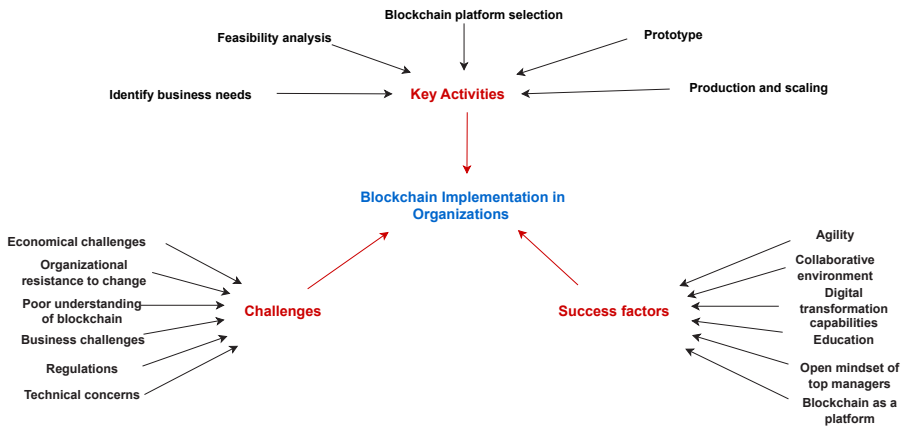


Figure 4.3: Emergence of core category, related categories and concepts by means of GT techniques

for blockchain technology in a specific environment has been outlined in literature [175]. Schulz et al. [175] highlighted that blockchain technology should not be seen as a *silver bullet* to address all problems.

- Two main dimensions were identified as significantly influencing the decision for blockchain technology: the legal dimension and blockchain-as-a-platform.
- The blockchain platform selection should be guided by foundational values of the organization, and a set of factors, e.g., network accessibility, transaction fees, consensus mechanisms, programmability, and community of developers.
- The majority of blockchain projects stop at the prototype/pilot stage. The core reason for this is that the decision to implement blockchain is pushed by the innovation team that tests out prototypes, instead of top management.
- The experts emphasized the organizational resistance to change, regarding the implementation of blockchain technology and elaborated on three main factors that may influence the resistance: innovation-production gap, conservative management, and centralized mentality.
- The following set of challenges were identified: costs (transaction fees, legal costs, cost of shifting into another BC platform, development, and infrastructure costs), fear of regulation and confusion regarding regulations, and technical issues (interoperability, scalability, and security).
- The following success factors were uncovered: being agile in order to be able to identify problems in a timely fashion and prioritize them,

4. Research Findings

involving customers in the early stages of the blockchain implementation process throughout all the meetings with blockchain experts/startups, establishing innovation ecosystems, as the result of collaboration between different parties to create new value streams. Moreover, it was found out that the blockchain ecosystem is characterized by decoupled collaboration that take the form of alliance structures, or public-private partnerships, and the experts outlined the organizational need to train personnel to get blockchain knowledge in-house and to enhance education throughout the organization in terms of how to identify blockchain use cases, the business value that blockchain can bring, and technical education to develop blockchain solutions.

- Additional success factors mentioned by the experts were to approach blockchain with an open mindset, to think big, and digitize larger chunks of the process that include different parties.

These findings were useful to improve the initial blockchain-enabled framework for RT, as depicted in Figure 4.4. Each component of the framework is explained in detail in P5.

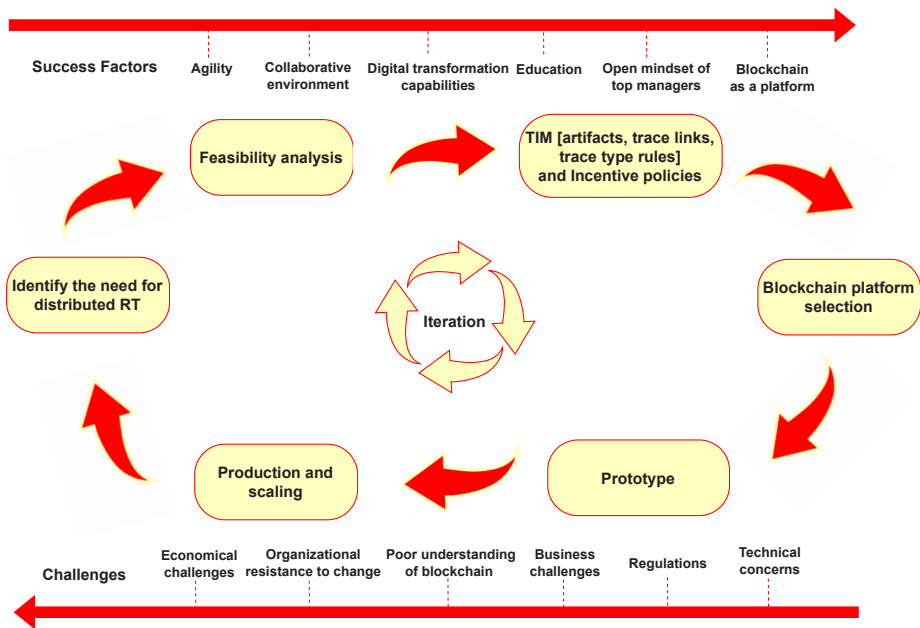


Figure 4.4: Framework to implement blockchain for RT in organizational settings

Contributions: This study contributes to the existing knowledge and practice regarding blockchain technology in two directions:

- (i) outlines the activities involved in implementing blockchain in organizations, along with associated challenges and success factors. The findings of

this exploratory study can be useful for researchers and practitioners interested in blockchain technology, increase the understanding of this technology by providing a holistic organizational approach and encourage the implementation of blockchain-based solutions, particularly in software-oriented organizations.

- (ii) enhances the quality and practicality of the initial framework by incorporating relevant categories grounded in empirical findings.

4.6 Paper VI: A Neural Blockchain for Requirements Traceability: BC4RT Prototype

S. Demi, R. Colomo-Palacios, M. Sánchez-Gordón, C. Velasco, and R. Cano, "*A Neural Blockchain for Requirements Traceability: BC4RT Prototype*," in Systems, Software and Services Process Improvement: 29th European Conference, EuroSPI 2022, Salzburg, Austria, August 31–September 2, 2022, Proceedings, pp. 45–59, Springer, 2022.

Purpose: As discussed in the Background section (See Chapter 2), there is a vast number of traceability studies that propose different tools and technologies to solve specific RT challenges. However, despite the substantial knowledge in this topic, the traceability community has highlighted that achieving full traceability in complex, large-scale and interorganizational software projects remains an open challenge [10, 28]. The goal of the sixth study of this thesis [7] is to propose and implement a blockchain-enabled prototype for the trustworthy management and traceability of software artifacts in interorganizational software projects based on the proposed framework in the previous study (P5). This prototype provides SDLC stakeholders with a holistic and reliable view of software artifacts, their changes and trace links. The enhanced visibility on the SDLC may potentially improve communication, coordination and trust among stakeholders, and consequently enhance software development efficiency and quality.

Prototype proposal and implementation: The prototype represents the implementation of the conceptual framework proposed in P5. The prototype was tested by means of a use case that considered four job roles: requirements manager, developer, tester, and customer. It is worthy to note that the prototype assumed a lack of trust among SDLC participants located in distributed settings. In this context, blockchain can serve as a secure software artifacts repository that ensures reliability, transparency, trust, traceability, and auditability. Each of the users (stakeholders) can carry out different operations. For instance, requirement managers can create new projects and requirements for each project. The projects, requirements, the timestamp of when requirements were created, contributor name, and current status “created” is also stored on blockchain. In addition, requirements managers can change requirements by updating their version, description or short name. This event changes the current status from “created” to “changed”.

4. Research Findings

For each requirement, developers can register related source code artifacts, and consequently the current status changes into “implemented”. Testers can register test cases and test results for each related requirement and the current status changes from “implemented” to “tested”. Finally, the customer can view all the changes of software artifacts and can track the lifecycle of each requirement. This prototype was the result of the implementation of the framework proposed in P5 that required the selection of a P2P platform that ensures efficiency, scalability, and security (See Section 2.1.4). Therefore, NDL ArcaNet was chosen as a collaborative P2P network that fulfils all the requirements of the use case and developed by the company that contributed to this PhD project, named ByEvolution Factory. A detailed description of this platform can be found in Chapter 2.

The blocks’ structure in Figure 4.5 depicts the creation of the project token with the following attributes: project token code, project domain, project name, token password, and additional information about the specific project. Each project token consists of a set of requirement tokens with the following attributes: ULID code of the requirement token, ULID code of the project that points to the parent project token, password of the requirement token, domain, requirement version, current status of requirements (i.e., created, changed, implemented, and tested), description and short name of the requirement, timestamps of when the requirement was created and updated, the contributor who carried out a specific operation on the token, flags (i.e., implemented or tested), related source code and test cases files.

The emission of the requirement token creates Block #1 which contains metadata, signatures of the previous block, content of the requirement token, signers of the next block as a combination of trusted and random nodes. The structure of consequent blocks is similar to the first block with the only difference that they store token changes, instead of the whole token content. The genesis block is provided randomly by the other network nodes. Finally, the user interface of the prototype is explained thoroughly in [68]. The feasibility of the prototype is validated by using the dataset of an electronic health records application, named iTrust in P6, and evaluated by SE experts in P7.

Contributions: This study contributes to the existing knowledge about blockchain and requirements traceability by proposing and implementing a blockchain-based prototype. This is the first prototype that enables SDLC stakeholders to keep track of who/how/when and by whom requirements were created, updated, implemented and/or tested in a trustworthy manner. In addition, the study explained a novel P2P collaborative platform that relies on the concept of neural distributed ledgers, named ArcaNet. This is one of the first published studies that leveraged the benefits of this platform: security, decentralization, scalability and performance efficiency. Although the uses of the NDL ArcaNet platform are currently unexplored, these benefits can be exploited in other domains, such as e-commerce and Internet-of-Things.

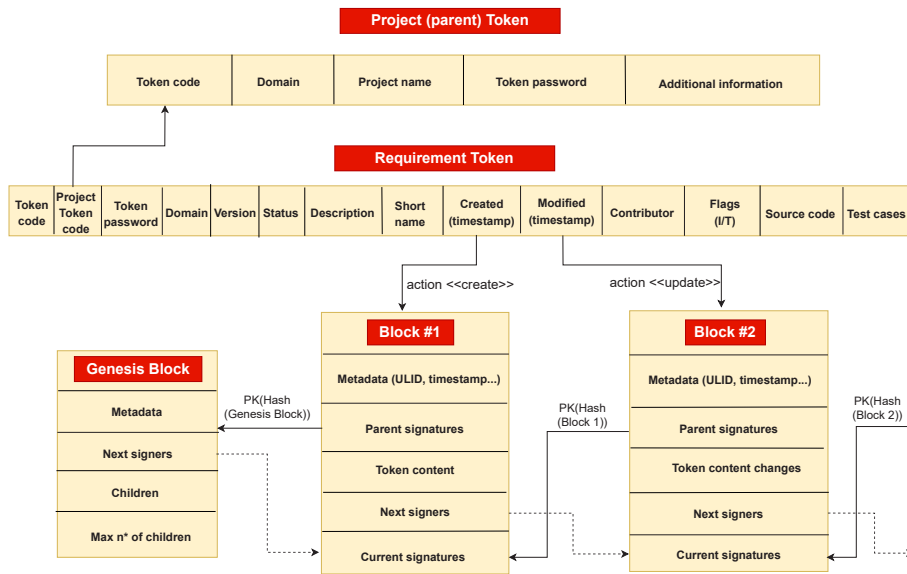


Figure 4.5: Underlying blocks' structure, published in [7]

4.7 Paper VII: Trustworthy and Collaborative Requirements Traceability: Validation of a Blockchain-enabled Framework

S. Demi, M. Sánchez-Gordón, and R. Colomo-Palacios, "*Trustworthy and Collaborative Requirements Traceability: Validation of a blockchain-enabled framework*", Submitted to Journal of Software: Evolution and Process.

Purpose: This study built on top of the previous studies of this PhD project. The paper focused on validating the proposed framework and prototype by using SE experts' judgement.

Research approach: This study adopted a qualitative data collection approach based on semi-structured interviews with SE experts. The interviews lasted for 45 minutes at maximum and were carried out through Zoom upon interviewees' consent. A set of predefined questions can be accessed online [131]. Experts were selected by using purposive sampling strategy, as the main sampling strategy used in the SE field [133, 134]. The experts were selected with a broad experience in different SE subdomains, with a combined academic and professional experience. The majority of experts had over 10 years of experience and all of them had high-level education (10% master's degree, 90% PhD). The transcripts of interviews

4. Research Findings

were analysed by using the content analysis approach proposed by Elo and Kyngäs [5]. The three main phases of the content analysis process: preparation, organizing and reporting were explained in Chapter 3.

Findings: The data analysis process revealed three categories: experts' perceptions, blockchain-based software process improvement, and experts' recommendations. The main findings are presented in the following section:

- *Experts' perceptions.* The perceptions of the experts regarding the proposed framework and prototype were positive. The framework was evaluated as easy to use, innovative and useful for requirements traceability and change management.
- *Blockchain-based software process improvement.* The following SPI values emerged from the analysis of data:
 - *Business.* The experts outlined traceability as the main benefit of the proposed blockchain-enabled system. Other benefits mentioned by the experts are reliability, availability of information, transparency, enhanced communication between stakeholders, and code reusability. The experts highlighted that these benefits can potentially lead to time-saving and improved project quality. Furthermore, the main concern raised by the experts was the highly complex documentation of requirements and related software artifacts that the proposed system requires. While it is true that this may affect the efficiency of the project and be perceived as anti-agile software development, it is also true that documentation is necessary even in agile software development. In addition, the experts contributed to defining the following 4 archetypes of organizations that may be interested in the proposed framework: globally distributed organizations, organizations with contract-based projects, organizations in safety-critical domains, and regulators.
 - *People.* The experts revealed the difficulty in finding software engineers with deep knowledge in blockchain technology, therefore they advocated for training and upskilling employees in this topic.
 - *Change.* The experts pointed out the resistance to change of software engineers regarding the implementation of the proposed framework, due to their tendency to not document software artifacts, performance limitations of blockchain technology, and the underlying conservative nature of software engineers.
- *Experts' recommendations.* The main recommendation of the experts was the integration of the proposed framework with the variety of tools used throughout the software lifecycle. Details about other promising recommendations, such as the inclusion of smart contracts, final phase “evaluate”, and different presentation formats tailored to stakeholders can be found in the last paper of this thesis (P7).

Contributions: The paper contributed to the final stage of the PhD project that entailed the evaluation of the proposed framework and prototype. In addition, the paper provided interesting insights into how the proposal can be improved by outlining promising future directions. These future directions proposed by experts with a broad experience in SE may pave the way for further improvements of existing approaches or proposals of new approaches that take advantage of inherent benefits of blockchain technology to address existing issues in global software engineering.

Chapter 5

Discussion

“Research is seeing what everybody else has seen and thinking what nobody else has thought”.
(Albert Szent-Györgyi)

In this section, the answers to the research questions formulated in Section 1.2 are discussed in relation to relevant previous literature. Moreover, the main contributions to theory and practice, and limitations of the thesis are discussed. Finally, the chapter presents promising research directions that may encourage further research efforts in advancing the blockchain-oriented SE field, with a particular emphasis on RT.

5.1 Synthesis of Findings

5.1.1 RQ1: What are the Challenges of Implementing Requirements Traceability?

Over the last three decades, the RE community has paid increasing attention to the RT topic. Nair et al. [102] analysed the trend of RT studies published in the Proceedings of the International RE conference. Unsurprisingly, the findings revealed an increasing interest of researchers in automated RT, and less interest on proposing new approaches for traceability maintenance. Overall, the authors concluded that traceability research is achieving maturity in terms of increased use of empirical methods, such as experiments.

However, automated traceability introduces new challenges. Our systematic literature review performed on 70 recent primary studies within the RT field revealed two core directions of traceability challenges research (P2): (i) identifying and/or addressing technical challenges related to traceability creation and maintenance, and (ii) identifying and/or addressing challenges attributed to the human factor in the tracing process. According to the findings of P2, the most frequently identified and/or addressed challenge was the low accuracy of traceability recovery methods (40%, 28 out of 70). This finding is in line with other literature reviews in the field, for instance Wang et al. [20] found out that the majority of research efforts were aimed at enhancing automation and trustworthiness of trace links. In addition, the findings of P2 indicated that human related challenges were addressed by only 25% of the studies. This finding suggested that human factors have been overlooked by the traceability community. This is not surprising, as human factors have often been overlooked in the SE domain [176].

Furthermore, the analysis of the primary studies in P2 revealed that the majority of traceability studies were published in conference proceedings. This

can be explained by the increasing presence of traceability research in the International RE conference, as demonstrated by Nair et al. [102]’s literature review. In fact, 32% of the conference studies in P2 were published in the International RE conference. This finding may imply that a significant number of traceability studies have been performed by the RE community which is also confirmed by Winkler and Pilgrim [31]. Finally, the analysis of the primary studies indicated that most of the studies adopted experiments as their main empirical method, whereas case studies and surveys were the least used methods. This finding is in line with the results of the literature review on RT carried out by Nair et al. [101].

5.1.2 RQ2: How can Blockchain Technology be used in the Software Engineering Domain?

The preliminary search on blockchain-enabled SE suggested an increasing number of studies focused on addressing SE issues in blockchain-based software, e.g., Vacca et al. [32] (For additional studies, see Section 2.3.1), but a limited number of studies that use blockchain to address issues in the software industry. While accessing a comprehensive overview of previous blockchain-based approaches in SE was important within the scope of this thesis, our search revealed only one related study carried out in 2019 by Tariq and Colomo-Palacios [33]. The findings of this systematic mapping study indicated that blockchain can be used to ensure trust among different parties in outsourced software development. For instance, a blockchain-based platform can enable software requestors to upload tasks and respective rewards and enable interested developers to upload code and get rewards once all tests pass in a decentralized and reliable manner.

In addition, Tariq and Colomo-Palacios [33]’s study emphasized the potential of smart contracts in verifying SE tasks automatically without the need for human intervention. The authors also outlined that blockchain can be used to trace third-party components added by developers and verify the compliance to license policies, as often team leaders are not aware of such components that can impact software quality and at a larger scale can affect the reputation of the company. While this study provided us with a better understanding of the potential applications and benefits that blockchain can bring to the software engineering landscape, it is limited to research conducted up to 2018. Given that blockchain research is rapidly evolving and new approaches are emerging even in the SE field, a more recent overview on the topic was considered of high value for advancing this PhD work.

The findings of the systematic mapping study conducted in P3 suggested that blockchain technology, given its inherent properties of decentralization, immutability, integrity, and non-repudiation (See Section 2.1.2) can be used to address SE issues in the following 4 directions:

1. *Decentralization.* Centralized systems that are commonly used today, such as GitHub, Travis CI, cloud-based package managers are prone to single point of failure and compromise issues [111]. To address such

issues, blockchain-based systems that offer decentralization, availability and resilience can be used.

2. *Ensure trust, visibility, and facilitate analysis.* Blockchain can be used as the backbone of the software lifecycle ecosystem that stores all relevant events, software artifacts, and their metadata in a trusted manner. This can be beneficial in enhancing visibility of the entire software lifecycle which can in turn enhance trust and collaboration among stakeholders in distributed settings. The full and holistic access on software-related events and artifacts facilitates different types of analysis, such as provenance and identity assessment, auditability, and compliance assessment.
3. *Secure sharing.* Blockchain can be used to enable collaborating parties to share software artifacts in a secure and trusted manner, as unauthorized attempts to access, update or transfer software artifacts can be detected. The immutability and integrity of software artifacts is ensured, once they are stored on blockchain.
4. *Automation.* Smart contracts can be used to automate a set of SE activities that usually require human rationale. For instance, smart contracts can be used to automatically verify if the implementation of requirements passed acceptance criteria, verify the compliance to regulations, policies, and best practices, and offer payments to software engineers in an automatic fashion. Automation by means of smart contracts was also unveiled in the systematic mapping study carried out by Tariq and Colomo-Palacios [33].

The findings in P3 revealed a limited number of prototypes and proof-of-concepts in the field, and the absence of blockchain-enabled SE applications implemented in organizational settings. In fact, this is not surprising, given the novelty of the topic (the first paper that used blockchain in SE was published in 2015 [110]), the nascent stage of blockchain technology, along with its unresolved challenges [107]. Furthermore, software engineers may argue that the concept of trust enabled by blockchain technology goes against the concept of trust enabled by agile software development. On one hand, one of the main principles of agile software development is the lack of control of the work performed and reliance on trust, transparency, communication and collective responsibility [79]. On the other hand, blockchain has been considered “trustless” [177], as it does not necessitate trusting participants, rather trusting the system which is secure, due to the combination of technologies explained in Chapter 2. Werbach [178] referred to the trust notion enabled by blockchain as “trustless trust”, meaning achieving collective trust on the underlying foundation of mutual distrust. While more research is needed in this regard, in this thesis the concepts of trust enabled by blockchain and agile software development are not perceived as contradicting, rather as complementary.

Furthermore, the findings indicated that the most used blockchain platforms in SE studies were Ethereum and Hyperledger Fabric. This is not surprising, given their capability to run code in the form of smart contracts or chain code.

The main difference between these platforms lies in network accessibility [179], therefore they can be used in different SE use cases discussed in P3. For instance, Ethereum can be used in the open-source ecosystem to manage packages and their dependencies on a public, transparent ledger which can facilitate code reuse [180], or in crowdsourcing projects where software development and testing tasks are delegated to the general public. On the other hand, Hyperledger Fabric, as a platform with permissioned accessibility, can be more suitable than Ethereum in intraorganizational and interorganizational software projects in which collaborating entities, often distributed, are known and they need to share software artifacts and related information securely [181]. More empirical evidence on the benefits of using blockchain in SE, alignment of blockchain principles with principles of software development practices, e.g., agile software development, and discussion on relevant blockchain platforms for SE use cases can encourage practitioners to invest in implementing blockchain technology.

5.1.3 RQ3: Is it possible to build a blockchain-enabled framework for decentralized and trustworthy requirements traceability that can be used by organizations in interorganizational software projects?

The initial framework for RT enabled by blockchain technology is described in P4 and leverages conventional blockchain platforms to store software artifacts metadata and trace links reliably. These platforms have been used by other SE researchers, for instance Bose et al. [118] used Ethereum to test their blockchain-enabled framework for software provenance, named Blinker. However, as stated by Bose et al. [118], developing blockchain applications for data-intensive use cases is challenging, due to the cost of storing data on Ethereum. This issue can be addressed by storing only software artifacts metadata and the cryptographic hashes of the data on blockchain, while the full data can be stored on off-chain databases that are immutable and distributed, such as IPFS. Although a similar approach was followed in P4, the off-chain storage of artifacts limits the ability to keep track of these artifacts and their changes in a trustworthy manner which is the ultimate goal of this PhD thesis.

The framework proposed in P4 was based on literature relevant to the topic, thus in order for the framework to be adopted by organizations, it is necessary to refine it by incorporating blockchain experts' judgement (P5). Since blockchain technology has been considered a disruptive and cutting-edge technology [182], it is not surprising that research leveraging its transformative potential is on the rise [183]. However, despite the hype on the topic, our research revealed that most of the studies take a technical perspective on issues, such as scalability, security and performance efficiency of blockchain systems, rather than an organizational process perspective. This could be a plausible explanation about why most of blockchain use cases fail and do not surpass the prototype stage. Given that this PhD thesis aims to provide software organizations with guidelines for building useful and usable blockchain-based systems, it is important to understand how

blockchain-enabled systems are implemented in organizational settings. To date, there are only a few studies that provide an organizational perspective of engagement with blockchain technology. Beck and Müller-Bloch [184] carried out a case study in an international bank that engages with blockchain in the different phases of discovery, incubation, and acceleration. The findings of this study suggested competencies needed to enable the transition from discovery to incubation and acceleration. In addition, the authors outlined the importance of collaboration on both intraorganizational and interorganizational level to fully leverage the potential of blockchain initiatives.

Dozier and Montgomery [185] unveiled the innovation evaluation process adopted by organizations to evaluate blockchain technology by using a grounded theory approach on a set of 19 interviews carried out in 12 U.S. financial organizations. Findings suggested the emergence of the proof-of-value model which consists of three phases: understanding the value that blockchain can bring, organize internally and externally to find value and test use cases to prove value. Ostern et al. [186] discovered four different types of blockchain approaches: the strategist, experimenter, implementer and observer, and their respective innovation phases: assimilation, implementation, adoption, and comprehension. While these approaches are focused on pre-adoption phase of blockchain technology, for instance Ostern et al. [186] focuses on how organizations make sense of blockchain technology, the implementation process of this technology remains vague and unexplored.

The findings in P5 indicated a set of activities that compose the implementation process of blockchain in organizational settings. The first activity is the identification of business needs and requirements. The main takeaway in this activity is that requirements should drive the choice of the technology and not vice versa. Therefore, blockchain should be implemented only if it addresses a problem that the organization faces. This is not trivial, as recent literature reported that blockchain is often perceived as a solution looking for a problem [185]. The identification of business needs should be followed by a feasibility analysis process on whether blockchain is the most appropriate solution compared with alternative solutions. In this regard, the results indicated two main dimensions: the legal dimension and blockchain as a platform that are explained in P5. In addition, the experts mentioned and elaborated on the selection process of blockchain platforms. This process was considered challenging by blockchain experts, due to the ever-increasing number of blockchain platforms which has been also mentioned in literature [187].

Despite the difficulty, the experts elaborated on a set of factors that should be considered when selecting blockchain platforms, such as the alignment with foundational values of the organization, network accessibility, transaction fees, consensus mechanisms, programmability, and community of developers. Interestingly, the findings suggested that in most cases, the implementation of blockchain technology in organizations is not pushed by top management, but it emerges from the innovation team that tests out prototypes. In line with that approach, Dozier and Montgomery [185] discovered that blockchain is tested in organizations through prototypes and proof of concepts in order to identify the

business value of implementing such technology, maximize the understanding and reduce possible risks.

Furthermore, the findings in P5 revealed that shifting blockchain prototypes into production occurs in rare cases. The limited number of implementations of blockchain in organizational settings has been confirmed by previous studies, e.g., [183]. In this regard, Flovik et al. [183] outlined the lack of empirical studies that investigate the reasons of limited implementations of blockchain technology. The blockchain experts in P5 suggested that one of the main reasons can be the innovation-production gap, meaning that those working with real products are often not included in the initial stages of the project, thus they show resistance when taking over the project from the innovation team. In fact, the implementation of blockchain is often not pushed into the organization by the top management, but by the innovation team. Other categories and concepts, and their detailed descriptions can be found in P5.

The findings in P5 contributed to refining the framework and enhancing its adoption. Although the framework was refined, the limitations of conventional blockchains mentioned in the first paragraph of this section and in P4 need to be addressed. These limitations were addressed in P6 with the implementation of a prototype that used NDL platforms to enable the secure storage of software artifacts of any type and size in a decentralized manner, while retaining scalability and performance efficiency (See Chapter 2). Researchers are recommended to leverage the potential of such platforms to process massive data, e.g., Internet-of-Things use cases can benefit from the security and efficiency of neural distributed ledgers.

Finally, the proposed framework and prototype were assessed by SE experts with a long experience in academia and/or industry in P7. While the use of blockchain technology for RT is an unexplored topic, software engineers' perception of the proposal was positive, particularly regarding the impact that blockchain can have on enhancing traceability, visibility and transparency of the software lifecycle. As outlined by experts in P7, blockchain technology can be beneficial, especially in regulated domains and projects based on contractual agreements between project partners. This finding is in line with previous studies that highlighted the importance of RT in safety-critical systems [14], and the value of traceability to prove the correctness of solutions, thus avoid costly disputes in projects based on contracts [30].

However, the experts highlighted a few concerns regarding the use of the prototype. The main concern lies in the high effort required by software lifecycle stakeholders to document requirements, related artifacts, and their changes. This high effort can lead to reluctance of stakeholders in using the proposed system. In fact, the high effort leading to reluctance has been acknowledged by the traceability community. Previous studies outlined that developers consider traceability an effort-intensive task to be performed in addition to their main work tasks [14, 28, 31], and managers consider traceability costly and its benefits invisible [14]. Furthermore, the heavyweight documentation of requirements and related artifacts may contradict to some extent the Agile principle of working software over comprehensive documentation [188]. While traceability can be

perceived as unnecessary in agile projects, traceability researchers, such as Cleland-Huang [189] pointed out that agile practices are increasingly leveraged in distributed, large-scale, and commonly in safety-critical projects. As confirmed by the experts in P7, traceability is of utmost importance in such projects, despite the high costs and efforts required to create and maintain trace links.

5.2 Contributions

Section 1.4 presented the main contributions of the thesis in relation to associated papers and research questions addressed. The contributions of the thesis to both theory and practice are discussed in the following sections.

5.2.1 Contributions to Theory

The thesis contributes to theory related to blockchain technology, SE with a particular emphasis on RT, and their interrelation. With regards to blockchain technology, the thesis uncovered the implementation process of blockchain in organizational settings, in terms of iterative key activities, success factors and challenges (P5). The findings in P5 contributed to explaining the organizational resistance to change when implementing blockchain technology through the following factors: innovation-production gap, centralized mentality, and conservative management. These findings can enhance the understanding of the implementation process of blockchain technology in organizational settings which is a research area with limited empirical evidence, to date [174, 190]. Moreover, the thesis contributed to the RT literature by providing a comprehensive and holistic overview of challenges that hinder the implementation of requirements traceability in practice (P2). With regards to the interrelation between blockchain and SE, the thesis contributed to existing knowledge by publishing the first systematic mapping journal study that unveiled blockchain use cases in SE and the benefits that blockchain technology can bring to the SE domain. This review may trigger novel use cases that leverage the potential of blockchain platforms to address SE issues.

From the research methods perspective, the thesis contributed to emphasizing the importance of qualitative research methods in SE. One may argue that qualitative methods, such as GT and content analysis are not suitable for technical fields, as they tend to focus on humans' experience, opinions and feelings. However, technology is created by and for humans, therefore their perspective is essential in designing usable and useful technological solutions. While qualitative methods, such as GT are becoming increasingly popular in SE, SE researchers may not be aware of its historical development and use [139]. This thesis consists of a GT study (P5) which was reported in detail for transparency and replicability purposes. As advocated by Stol et al. [139], well-conducted GT studies can lead to significant contributions to the SE field.

Finally, to the best of our knowledge, the papers that comprise this thesis are the first published studies that proposed, designed, developed, and

evaluated a blockchain-enabled framework and prototype for trustworthy RT in interorganizational software projects. Although the topic was first proposed in the studies published within the scope of this PhD thesis, it has encouraged the work of other researchers in the field. For instance, in 2022, Farooq et al. [191] proposed a software requirements engineering model enabled by blockchain technology. According to the authors, this model addressed requirements ambiguities issues, communication gap between stakeholders, along with traceability and transparency issues. During the same year, Marjanović et al. [192] proposed a blockchain-enabled model that keeps track of compliance with security requirements by providing reliable visibility of secure software development lifecycle metrics to authorized auditors. These studies indicate promising dimensions of the blockchain-enabled RT topic that are worthy to be explored.

5.2.2 Contributions to Practice

The thesis contributes to practice by designing two main approaches (P4, P6) that enable decentralized, yet trustworthy RT by using conventional blockchain platforms and neural distributed ledger platforms. The framework, proposed in P4 and improved in P5, was designed in such a way that it can be adopted by organizations and tailored to their specific business needs and requirements. As mentioned in P4, P5 and in the BOSE literature [104], the implementation of BOSE approaches requires expertise in both fields of blockchain and SE. Therefore, the incorporation of blockchain experts and SE experts' judgment led to useful insights into how to improve the prototype, and potential benefits and challenges that can drive or hinder its implementation, respectively. The empirical evidence provided by experts can serve as best-practice and guide the implementation of blockchain technology for trustworthy RT in interorganizational software projects. It is worthy to note, that the categories grounded in empirical data in P5 are of general and domain-agnostic nature, hence they can guide the implementation of blockchain-based systems in different domains. Finally, a blockchain-enabled RT prototype was developed and its feasibility was validated by using the dataset of an open-source electronic health records application, named iTrust, and SE experts' judgement. The development of such prototype can encourage innovation teams of interested organizations to build similar prototypes and evaluate their feasibility.

5.3 Limitations

In this section, the main limitations of the thesis are summarized. First, the novelty of the blockchain-enabled RT topic can be considered a limitation, as it was not possible to build on top of previous approaches. This means that the PhD student had to find alternative ways to acquire knowledge and trigger ideas on the topic. To address this limitation, the student dedicated the first year of the PhD study to review systematically RT challenges, and blockchain use

cases in SE. Even though, the blockchain-enabled SE field is not yet mature, the reviews (P2, P3) provided the student with a good understanding of the topics of interest, which in turn enabled the design, development and evaluation of the proposed framework and prototype.

Second, the thesis incorporates judgement of 20 blockchain and SE experts to refine and evaluate the proposals. It is worthy to outline the sampling size limitation, particularly when applying GT analysis techniques in P5. While it is true that GT aims to generate a theory grounded in empirical data which usually requires a higher number of respondents, it is also true that our goal in P5 was not to generate a theory, rather to define categories that can be used to enhance the practicality of our proposed framework. Similarly, the goal of the interview study in P7 was not to achieve generalizability, rather to evaluate the strengths and limitations of the prototype.

Third, the main limitations of the proposed prototype were outlined by SE experts in P7. The main limitations of the prototype are high efforts required to document requirements, related software artifacts, their metadata and changes, and the lack of integration with existing tools used throughout the software lifecycle. These limitations should be addressed to encourage the adoption of the prototype by organizations interested in achieving trustworthy RT in their interorganizational software projects. Finally, it is worthy to mention evaluation limitations of the prototype in organizational settings. Although, the prototype was assessed by SE experts in P7, and its utility has been demonstrated by using software artifacts of an electronic health records application in P6, the prototype has not been evaluated with real-world interorganizational software projects, due to time constraints of this PhD project. Such evaluation would also require the development of plugins to integrate the blockchain-enabled proposal with existing tools used throughout the software development lifecycle, which was not feasible, due to limited resources. This limitation paves the way for future research work.

5.4 Future Work

This section discusses promising future research avenues in three dimensions that are consistent with the research questions formulated in Section 1.2:

- (i) *RT field*. The systematic literature review in P2 revealed the following underexplored areas that can be addressed in future research:
 - *Distributed traceability*. The shift towards the distributed development paradigm and the increased need for collaboration across organizational boundaries call for approaches that enable distributed traceability management. As mentioned in Chapter 1, interorganizational collaboration introduces confidentiality constraints which make artifacts with sensitive information inaccessible, for instance artifacts that include intellectual property are not shared by OEMs in the automotive domain [14]. Since the collaborating entities cannot

access the entire set of artifacts, they have to rely on the traceability information provided by the project partner. In such a context, it is important to ensure that the traceability information can be trusted even across organizational boundaries [30] which is not trivial, given the issue of mistrust in interorganizational software projects [193]. New disruptive technologies, such as blockchain technology can contribute to address these trust issues in interorganizational software projects [103].

- *Human factors.* As mentioned in Chapter 5, human factors have been overlooked in the traceability research. With regards to human factors, two promising directions for future research can be suggested: identify factors that affect the performance of analysts during the traceability vetting process and how to assist analysts in terms of tool support, e.g., incorporating tagging or contextual information into tracing tools, and propose approaches to incentivize practitioners to create and maintain quality trace links, e.g., incorporation of gamification features into tracing tools.
 - *Traceability in agile and continuous SE.* Conventional RT techniques are not appropriate in the context of agile and continuous SE, due to the characteristics of such environments, such as the absence of requirements documentation, continuous integration, testing and delivery. While Wang et al. [20] discovered only one study, the SLR in P2 identified 6 additional studies on the topic of agile-oriented traceability. In this regard, more studies that propose lightweight RT approaches are needed to support the increasing popularity of agile and DevOps practices.
 - *More exploratory studies.* The low number of exploratory studies on traceability practices pinpoints the need for more empirical evidence on how traceability is performed in organizational settings, and what are the needs and challenges of implementing traceability. This evidence can increase the understanding of traceability as a quality attribute of the software development lifecycle, and consequently it can motivate low-end users of traceability to adopt traceability practices.
- (ii) *Blockchain-oriented SE field.* The systematic mapping in P3 suggested the following future research directions regarding the blockchain-oriented SE field:
- Propose approaches that use smart contracts for the automatic validation of software requirements and design against pre-defined criteria. A lack of such approaches was observed in the systematic mapping performed in P3.
 - While benefits that blockchain can bring to the SE domain have been mentioned in literature [110], there is a need to investigate holistically the effects of blockchain technology on software development efficiency and software quality.

- As mentioned in Chapter 5, agile principles may contradict blockchain principles at first glance. Hence, interested researchers can contribute to enhancing the understanding of this complex relation by mapping, comparing, and aligning blockchain principles with agile principles.
 - Another interesting future direction can be to map blockchain platforms and their features with SE use cases, to guide software engineers when choosing the most suitable blockchain platform for their specific use case.
 - In order to leverage the benefits of blockchain in the context of SE, professionals with deep knowledge in both fields are necessary. Thus, future efforts should be devoted to addressing the need for professionals competent in both blockchain technology and SE.
 - Finally, researchers are recommended to explore and investigate emerging blockchain 4.0 platforms with regards to their applications in SE, as these platforms incorporate AI features which are expected to expand the benefits of blockchain technology on SE practices [109].
- (iii) *Improvements of the proposed blockchain-enabled framework and prototype for RT.*

Time and resources limitations led to limitations of the proposed framework and prototype, as mentioned by SE experts (See Section 5.3). However, these limitations pave the way for promising future work directions. The main future work direction lies in the evaluation of the prototype in organizational settings. To do so, it is necessary to enable lightweight integration of the blockchain proposal with existing tools used throughout the software lifecycle, ranging from Rational DOORS to Jira and GitHub. This lightweight integration may encourage the adoption of blockchain technology by software engineers and may enable researchers to perform case studies on safety-critical systems to investigate the feasibility and benefits of the blockchain-enabled RT prototype. In addition, the experts in P7 provided recommendations that can be useful to enhance the prototype in future work, as follows:

- inclusion of change rationale. While the prototype keeps track of software artifacts' changes, it does not provide information of the rationale behind changes.
- incorporation of smart contracts. Smart contracts can be incorporated into the prototype, to automatically assess the implementation of requirements or to assess compliance with regulations, policies, and service-learning agreements.
- tailored representation of traceability information. The visualization of traceability information should be tailored to different stakeholders, for instance, limit technical information (e.g., ULID) presented to the customer who may not have a technical background. Moreover, the use of highlighting techniques was suggested to differentiate successfully tested requirements from requirements with failed test cases.

5. Discussion

- incentivization of software engineers to use the proposed system. The SE experts suggested the incorporation of gamification features and financial or social benefits for each trace link created by stakeholders.

In order to address sampling size limitations mentioned in Section 5.3, future efforts may be devoted to enhancing the prototype by means of questionnaires with a larger number of software engineers with experience in the development of safety-critical systems, relying on Pfleeger and Kitchenham [194]’s guidelines for surveys in SE. In addition, the practicality of the framework can be enhanced by replicating the grounded theory study in P5 with a larger number of blockchain experts and triangulation of data collection methods, e.g., observation and document analysis, in addition to interviews. New data may evolve and enhance the concepts and categories related to the implementation of blockchain in organizational settings.

Finally, future efforts can be devoted to addressing the gap in understanding the representation of traceability information stored on blockchain, despite studies that demonstrated significant advantages of graph-based traceability models [195, 196]. Although the uses of graph technologies in blockchain are still evolving, some organizations (e.g. BitExTract, Bitquery, BlockchainVis) already offer graph technologies and related visualization tools for blockchains. Gartner predicts that by 2023, graph technologies will facilitate decision-making in 30% of organizations around the world [197]. By 2025, Gartner predicts that graph technologies will enable 80% of analytics innovations, resulting in faster decision-making [198].

Chapter 6

Concluding Remarks

“The important is to never stop questioning [or learning]”.
(Albert Einstein)

This PhD thesis explored the use of blockchain technology to keep track of software requirements, their related software artifacts, and their changes, in a trustworthy manner. The thesis challenges the traditional centralized way of storing and sharing software artifacts, by proposing a blockchain-oriented framework that enables a decentralized, yet reliable traceability knowledge base. Ensuring this reliable traceability knowledge base is important, particularly in complex and large-scale software projects that often operate in safety-critical domains and rely on the collaboration of multiple stakeholders in distributed locations. In addition, these stakeholders may be third-party entities with malicious intentions, thus the authenticity of the software artifacts and traceability information provided by these parties cannot be trusted by default. In this regard, blockchain technology can be considered the most suitable technology to address such trust issues and to ensure that stakeholders work on the same reality. It is worthy to highlight that, to the best of our knowledge, this thesis consists of the first published studies that explored the application of blockchain technology for trustworthy RT.

A total of 7 scientific papers have been published within the three-year timeframe of this PhD project in international conferences and peer-reviewed journals. During the first year of the PhD program, the PhD student focused on reviewing the relevant literature in the following two fields of interest: blockchain technology and requirements traceability. Three papers were published during this period. The first paper (P1) presented the initial research plan and was presented in the Doctoral Symposium section of the 28th IEEE International Requirements Engineering Conference. The participation in this conference contributed significantly to the next research stages, as critical and relevant feedback was received from the top experts in the field. In addition to presenting the research plan, the PhD student was part of the organizing team of the conference, an experience that enabled the student to be part of the RE community and participate actively in relevant RE-related discussions. The second study (P2) focused on identifying and categorizing RT challenges, as reported by recent scientific studies in the field, and the third study (P3) aimed to unveil blockchain-enabled applications that address SE issues, along with the benefits that blockchain technology can bring to the SE field. These studies were deliberately conducted in the initial phases of the PhD program, to guide the next stages by triggering ideas about “why” and “how” blockchain technology can be used for RT.

6. Concluding Remarks

The findings of the review studies (P2, P3) were used to design the initial blockchain-enabled RT framework (P4) that relied on conventional blockchain platforms and the concept of smart contracts to register software artifacts' metadata retrieved by SDLC tools and trace links created by distributed stakeholders. The proposal was presented and discussed in the Emerging and Multidisciplinary Approaches to Software Engineering workshop, part of EuroSPI 2021 conference. The proposed framework was improved in the fifth paper (P5) by means of blockchain experts' judgement and it was published during the second year of the PhD program. The development and validation of the blockchain prototype for requirements traceability, named BC4RT were conducted during the third year of the PhD project. The prototype was implemented by using a neural distributed ledger platform, named ArcaNet, that enables a decentralized, secure, efficient, and scalable approach to store requirements and related software artifacts of any type and size, along with their changes. This approach provides stakeholders of the software lifecycle with a holistic, auditable, and tamper-proof view on what/how/when software artifacts were created and/or updated, which in turn enhances communication, and coordination and trust among stakeholders. Ultimately, these benefits may lead to enhanced software development efficiency and software quality. Finally, the prototype was evaluated by SE experts who provided interesting insights into strengths and limitations of the approach. Moreover, these experts recommended improvements of different components of the prototype that paved the way for promising future research avenues.

Future work efforts could be devoted to improving the prototype based on the SE experts' recommendations (See Chapter 5 and P7) and operationalizing the proposed approach in organizational settings. In order to operationalize the proposal, it could be interesting to integrate it with existing tools used throughout the software development lifecycle, e.g., Rational Doors, GitHub, and Jira, in a lightweight manner. The development of plugins to retrieve software artifacts' metadata, content and their changes from existing tools can automate their registration and consequently, relieve software engineers' work and allow them to allocate time for more profitable tasks. Finally, the prototype can be validated by experts with experience in the development of safety-critical systems and based on their feedback, it can be potentially implemented in real-life software projects.

Bibliography

- [1] M. Belotti, N. Božić, G. Pujolle, and S. Secci, “A vademecum on blockchain technologies: When, which, and how,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3796–3838, 2019.
- [2] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design science in information systems research,” *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.
- [3] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic mapping studies in software engineering,” in *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, pp. 1–10, 2008.
- [4] R. Hoda, J. Noble, and S. Marshall, “Using grounded theory to study the human aspects of software engineering,” in *Human Aspects of Software Engineering*, pp. 1–2, 2010.
- [5] S. Elo and H. Kyngäs, “The qualitative content analysis process,” *Journal of advanced nursing*, vol. 62, no. 1, pp. 107–115, 2008.
- [6] S. Demi, M. Sánchez-Gordón, and R. Colomo-Palacios, “A blockchain-enabled framework for requirements traceability,” in *Systems, Software and Services Process Improvement: 28th European Conference, EuroSPI 2021, Krems, Austria, September 1–3, 2021, Proceedings*, pp. 3–13, Springer, 2021.
- [7] S. Demi, R. Colomo-Palacios, M. Sánchez-Gordón, C. Velasco, and R. Cano, “A neural blockchain for requirements traceability: Bc4rt prototype,” in *Systems, Software and Services Process Improvement: 29th European Conference, EuroSPI 2022, Salzburg, Austria, August 31–September 2, 2022, Proceedings*, pp. 45–59, Springer, 2022.
- [8] O. C. Z. Gotel and S. J. Morris, “Out of the labyrinth: Leveraging other disciplines for requirements traceability,” in *2011 IEEE 19th International Requirements Engineering Conference*, pp. 121–130, 2011.
- [9] B. W. Boehm, “Software engineering,” *IEEE Trans. Computers*, vol. 25, no. 12, pp. 1226–1241, 1976.
- [10] O. Gotel, J. Cleland-Huang, J. H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, J. Maletic, and P. Mäder, “Traceability fundamentals,” *Software and systems traceability*, pp. 3–22, 2012.

- [11] O. C. Gotel and C. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of IEEE international conference on requirements engineering*, pp. 94–101, IEEE, 1994.
- [12] B. Ramesh, "Factors influencing requirements traceability practice," *Communications of the ACM*, vol. 41, no. 12, pp. 37–44, 1998.
- [13] O. Gotel, J. Cleland-Huang, J. H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, and J. Maletic, "The grand challenge of traceability (v1.0)," *Software and Systems Traceability*, pp. 343–409, 2012.
- [14] S. Maro, J.-P. Steghöfer, and M. Staron, "Software traceability in the automotive domain: Challenges and solutions," *Journal of Systems and Software*, vol. 141, pp. 85–110, 2018.
- [15] P. Mäder, P. L. Jones, Y. Zhang, and J. Cleland-Huang, "Strategic traceability for safety-critical projects," *IEEE software*, vol. 30, no. 3, pp. 58–66, 2013.
- [16] G. Regan, F. Mc Caffery, K. Mc Daid, and D. Flood, "Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model," *Computer Standards & Interfaces*, vol. 36, no. 1, pp. 3–9, 2013.
- [17] G. Regan, F. McCaffery, K. McDaid, and D. Flood, "Traceability-why do it?," in *Software Process Improvement and Capability Determination: 12th International Conference, SPICE 2012, Palma, Spain, May 29-31, 2012. Proceedings 12*, pp. 161–172, Springer, 2012.
- [18] S. Murugappan and D. Prabha, "Requirement traceability for software development lifecycle," *International Journal of Scientific & Engineering Research*, vol. 8, no. 5, pp. 1–11, 2017.
- [19] D. Kchaou, N. Bouassida, M. Mefteh, and H. Ben-Abdallah, "Recovering semantic traceability between requirements and design for change impact analysis," *Innovations in Systems and Software Engineering*, vol. 15, pp. 101–115, 2019.
- [20] B. Wang, R. Peng, Y. Li, H. Lai, and Z. Wang, "Requirements traceability technologies and technology transfer decision support: A systematic review," *Journal of Systems and Software*, vol. 146, pp. 59–79, 2018.
- [21] F. Blaauboer, K. Sikkel, and M. N. Aydin, "Deciding to adopt requirements traceability in practice," in *Advanced Information Systems Engineering: 19th International Conference, CAiSE 2007, Trondheim, Norway, June 11-15, 2007. Proceedings 19*, pp. 294–308, Springer, 2007.

-
- [22] L. Klimpke and T. Hildenbrand, "Towards end-to-end traceability: Insights and implications from five case studies," in *2009 Fourth International Conference on Software Engineering Advances*, pp. 465–470, IEEE, 2009.
- [23] D. Berry, R. Gacitua, P. Sawyer, and S. F. Tjong, "The case for dumb requirements engineering tools," in *Requirements Engineering: Foundation for Software Quality: 18th International Working Conference, REFSQ 2012, Essen, Germany, March 19-22, 2012. Proceedings 18*, pp. 211–217, Springer, 2012.
- [24] H. Saiedian, A. Kannenberg, and S. Morozov, "A streamlined, cost-effective database approach to manage requirements traceability," *Software Quality Journal*, vol. 21, pp. 23–38, 2013.
- [25] N. Niu, W. Wang, and A. Gupta, "Gray links in the use of requirements traceability," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 384–395, 2016.
- [26] S. Gayer, A. Herrmann, T. Keuler, M. Riebisch, and P. O. Antonino, "Lightweight traceability for the agile architect," *Computer*, vol. 49, no. 5, pp. 64–71, 2016.
- [27] R. Torkar, T. Gorschek, R. Feldt, M. Svahnberg, U. A. Raja, and K. Kamran, "Requirements traceability: a systematic review and industry case study," *International Journal of Software Engineering and Knowledge Engineering*, vol. 22, no. 03, pp. 385–433, 2012.
- [28] R. Wohlrab, E. Knauss, J.-P. Steghöfer, S. Maro, A. Anjorin, and P. Pelliccione, "Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture," *Requirements Engineering*, vol. 25, no. 1, pp. 21–45, 2020.
- [29] S. S. Yau and J. S. Patel, "Application of blockchain for trusted coordination in collaborative software development," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1036–1040, IEEE, 2020.
- [30] P. Rempel, P. Mäder, T. Kuschke, and I. Philippow, "Requirements traceability across organizational boundaries-a survey and taxonomy," in *Requirements Engineering: Foundation for Software Quality: 19th International Working Conference, REFSQ 2013, Essen, Germany, April 8-11, 2013. Proceedings 19*, pp. 125–140, Springer, 2013.
- [31] S. Winkler and J. von Pilgrim, "A survey of traceability in requirements engineering and model-driven development," *Software & Systems Modeling*, vol. 9, pp. 529–565, 2010.
- [32] A. Vacca, A. Di Sorbo, C. A. Visaggio, and G. Canfora, "A systematic literature review of blockchain and smart contract development: Techniques,

- tools, and open challenges,” *Journal of Systems and Software*, vol. 174, p. 110891, 2021.
- [33] F. Tariq and R. Colomo-Palacios, “Use of blockchain smart contracts in software engineering: A systematic mapping,” in *Computational Science and Its Applications–ICCSA 2019: 19th International Conference, Saint Petersburg, Russia, July 1–4, 2019, Proceedings, Part V 19*, pp. 327–337, Springer, 2019.
- [34] P. Thakkar, S. Nathan, and B. Viswanathan, “Performance benchmarking and optimizing hyperledger fabric blockchain platform,” in *2018 IEEE 26th international symposium on modeling, analysis, and simulation of computer and telecommunication systems (MASCOTS)*, pp. 264–276, IEEE, 2018.
- [35] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, “Solutions to scalability of blockchain: A survey,” *Ieee Access*, vol. 8, pp. 16440–16455, 2020.
- [36] B. Kitchenham, “Procedures for performing systematic reviews,” *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.
- [37] D. Tapscott and A. Tapscott, *Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world*. Penguin, 2016.
- [38] W. Mougayar, *The business blockchain: promise, practice, and application of the next Internet technology*. John Wiley & Sons, 2016.
- [39] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized business review*, p. 21260, 2008.
- [40] A. T. Sherman, F. Javani, H. Zhang, and E. Golaszewski, “On the origins and variations of blockchain technologies,” *IEEE Security & Privacy*, vol. 17, no. 1, pp. 72–77, 2019.
- [41] R. C. Merkle, “Secure communications over insecure channels,” *Communications of the ACM*, vol. 21, no. 4, pp. 294–299, 1978.
- [42] N. Szabo, “Smart contracts.” <https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html>, 1994. Accessed: 2022-03-17.
- [43] V. Buterin, “Ethereum whitepaper.” <https://ethereum.org/en/whitepaper/>, 2014. Accessed: 2021-02-01.
- [44] G. O. Karame, E. Androulaki, and S. Capkun, “Double-spending fast payments in bitcoin,” in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 906–917, 2012.
- [45] A. R. Sai, J. Buckley, B. Fitzgerald, and A. Le Gear, “Taxonomy of centralization in public blockchain systems: A systematic literature review,” *Information Processing & Management*, vol. 58, no. 4, p. 102584, 2021.

-
- [46] A. R. Sai, J. Buckley, and A. Le Gear, "Assessing the security implication of bitcoin exchange rates," *Computers & Security*, vol. 86, pp. 206–222, 2019.
- [47] A. E. Gencer, S. Basu, I. Eyal, R. Van Renesse, and E. G. Sirer, "Decentralization in bitcoin and ethereum networks," in *Financial Cryptography and Data Security: 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26–March 2, 2018, Revised Selected Papers 22*, pp. 439–457, Springer, 2018.
- [48] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun, "Is bitcoin a decentralized currency?," *IEEE security & privacy*, vol. 12, no. 3, pp. 54–60, 2014.
- [49] Q. DuPont, "Experiments in algorithmic governance: A history and ethnography of "the dao," a failed decentralized autonomous organization," in *Bitcoin and beyond*, pp. 157–177, Routledge, 2017.
- [50] D. Kondor, M. Pósfai, I. Csabai, and G. Vattay, "Do the rich get richer? an empirical analysis of the bitcoin transaction network," *PloS one*, vol. 9, no. 2, p. e86197, 2014.
- [51] A. R. Sai, J. Buckley, and A. Le Gear, "Privacy and security analysis of cryptocurrency mobile applications," in *2019 fifth conference on mobile and secure services (MobiSecServ)*, pp. 1–6, IEEE, 2019.
- [52] K. Sagar Bharadwaj, S. Dharanikota, A. Honawad, and K. Chandrasekaran, "Blockchain research and applications: A systematic mapping study," in *IC-BCT 2019: Proceedings of the International Conference on Blockchain Technology*, pp. 141–159, Springer, 2020.
- [53] D. Conte de Leon, A. Q. Stalick, A. A. Jillepalli, M. A. Haney, and F. T. Sheldon, "Blockchain: properties and misconceptions," *Asia Pacific Journal of Innovation and Entrepreneurship*, vol. 11, no. 3, pp. 286–300, 2017.
- [54] M. Swan, *Blockchain: Blueprint for a New Economy*. O'Reilly Media, 2015.
- [55] M. Xu, X. Chen, and G. Kou, "A systematic review of blockchain," *Financial Innovation*, vol. 5, no. 1, pp. 1–14, 2019.
- [56] R. Colomo-Palacios, M. Sánchez-Gordón, and D. Arias-Aranda, "A critical review on blockchain assessment initiatives: A technology evolution viewpoint," *Journal of Software: evolution and process*, vol. 32, no. 11, p. e2272, 2020.
- [57] J. Angelis and E. R. Da Silva, "Blockchain adoption: A value driver perspective," *Business Horizons*, vol. 62, no. 3, pp. 307–314, 2019.

- [58] R. Arenas and P. Fernandez, “Credenceledger: a permissioned blockchain for verifiable academic credentials,” in *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pp. 1–6, IEEE, 2018.
- [59] Hackernoon, “What will the fourth generation of blockchain look like?.” <https://hackernoon.com/what-will-the-fourth-generation-of-blockchain-look-like-daa5a4e90c59>, 2019. Accessed: 2022-04-10.
- [60] F. Tschorsch and B. Scheuermann, “Bitcoin and beyond: A technical survey on decentralized digital currencies,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.
- [61] G. Wood *et al.*, “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.
- [62] V. Buterin, D. Reijnders, S. Leonardos, and G. Piliouras, “Incentives in ethereum’s hybrid casper protocol,” *International Journal of Network Management*, vol. 30, no. 5, p. e2098, 2020.
- [63] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, *et al.*, “Hyperledger fabric: a distributed operating system for permissioned blockchains,” in *Proceedings of the thirteenth EuroSys conference*, pp. 1–15, 2018.
- [64] N. Vadgama, J. Xu, and P. Tascia, *Enabling the Internet of Value*. Springer, 2022.
- [65] C. Velasco, R. Colomo-Palacios, and R. Cano, “Neural distributed ledger,” *IEEE Software*, vol. 37, no. 5, pp. 43–48, 2020.
- [66] F. L. Benítez-Martínez, M. V. Hurtado-Torres, and E. Romero-Frías, “A neural blockchain for a tokenizable e-participation model,” *Neurocomputing*, vol. 423, pp. 703–712, 2021.
- [67] F. L. Benítez-Martínez, E. Romero-Frías, and M. V. Hurtado-Torres, “Neural blockchain technology for a new anticorruption token: towards a novel governance model,” *Journal of Information Technology & Politics*, vol. 20, no. 1, pp. 1–18, 2023.
- [68] “Byevolution creative company.” <https://byevolution.com/>. Accessed: 2022-06-08.
- [69] B.-M. Francisco Luis, N.-C.-U. Pedro Víctor, M.-M. Valentín, and R.-F. Esteban, “Blockchain as a service: A holistic approach to traceability in the circular economy,” in *Blockchain Technologies for Sustainability*, pp. 119–133, Springer, 2021.

-
- [70] O. Cico, L. Jaccheri, A. Nguyen-Duc, and H. Zhang, “Exploring the intersection between software industry and software engineering education—a systematic mapping of software engineering trends,” *Journal of Systems and Software*, vol. 172, p. 110736, 2021.
- [71] P. Naur and B. Randell, “Software engineering: Report of a conference sponsored by the nato science committee, garmisch, germany, 7th-11th october 1968,” 1969.
- [72] “Ieee standard glossary of software engineering terminology,” *IEEE Std 610.12-1990*, pp. 1–84, 1990.
- [73] SEBoK, “An overview of the swebok guide — sebok,.” https://sebokwiki.org/w/index.php?title=An_Overview_of_the_SWEBOK_Guide&oldid=66059, 2021. Accessed: 2023-01-17.
- [74] “Software engineering body of knowledge: Swebok.” <https://www.computer.org/education/bodies-of-knowledge/software-engineering>. Accessed: 2021-02-20.
- [75] C. Ebert, M. Kuhrmann, and R. Prikladnicki, “Global software engineering: Evolution and trends,” in *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, pp. 144–153, IEEE, 2016.
- [76] M. Niazi, S. Mahmood, M. Alshayeb, M. R. Riaz, K. Faisal, N. Cerpa, S. U. Khan, and I. Richardson, “Challenges of project management in global software development: A client-vendor analysis,” *Information and Software Technology*, vol. 80, pp. 1–19, 2016.
- [77] H. Holmstrom, E. Ó. Conchúir, J. Agerfalk, and B. Fitzgerald, “Global software development challenges: A case study on temporal, geographical and socio-cultural distance,” in *2006 IEEE International Conference on Global Software Engineering (ICGSE'06)*, pp. 3–11, IEEE, 2006.
- [78] D. Šmite, N. B. Moe, and P. J. Ågerfalk, “Fundamentals of agile distributed software development,” *Agility across time and space: Implementing agile methods in global software projects*, pp. 3–7, 2010.
- [79] O. McHugh, K. Conboy, and M. Lang, “Agile practices: The impact on trust in software project teams,” *IEEE software*, vol. 29, no. 3, pp. 71–76, 2011.
- [80] R. Vallon, B. J. da Silva Estácio, R. Prikladnicki, and T. Grechenig, “Systematic literature review on agile practices in global software development,” *Information and Software Technology*, vol. 96, pp. 161–180, 2018.
- [81] K. Dikert, M. Paasivaara, and C. Lassenius, “Challenges and success factors for large-scale agile transformations: A systematic literature review,” *Journal of Systems and Software*, vol. 119, pp. 87–108, 2016.

- [82] M. Berntzen, R. Hoda, N. B. Moe, and V. Stray, “A taxonomy of inter-team coordination mechanisms in large-scale agile,” *IEEE Transactions on Software Engineering*, vol. 49, no. 2, pp. 699–718, 2022.
- [83] I. Sommerville and J. Ransom, “An empirical study of industrial requirements engineering process assessment and improvement,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 14, no. 1, pp. 85–117, 2005.
- [84] D. Fucci, C. Palomares, X. Franch, D. Costal, M. Raatikainen, M. Stettinger, Z. Kurtanovic, T. Kojo, L. Koenig, A. Falkner, *et al.*, “Needs and challenges for a platform to support large-scale requirements engineering: A multiple-case study,” in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 1–10, 2018.
- [85] D. M. Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, *et al.*, “Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice,” *Empirical software engineering*, vol. 22, pp. 2298–2338, 2017.
- [86] R. R. Lutz, “Analyzing software requirements errors in safety-critical, embedded systems,” in *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*, pp. 126–133, IEEE, 1993.
- [87] T. Hall, S. Beecham, and A. Rainer, “Requirements problems in twelve software companies: an empirical analysis,” *IEE Proceedings - Software*, vol. 149, no. 5, pp. 153–160, 2002.
- [88] D. M. Fernández and S. Wagner, “Naming the pain in requirements engineering: design of a global family of surveys and first results from germany,” in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, pp. 183–194, 2013.
- [89] D. M. Fernandez, “Supporting requirements-engineering research that industry needs: The napire initiative,” *IEEE Software*, vol. 35, no. 01, pp. 112–116, 2018.
- [90] S. Jayatilleke and R. Lai, “A systematic review of requirements change management,” *Information and Software Technology*, vol. 93, pp. 163–185, 2018.
- [91] D. E Damian and D. Zowghi, “Re challenges in multi-site software development organisations,” *Requirements engineering*, vol. 8, pp. 149–160, 2003.
- [92] M. A. Akbar, A. Alsanad, S. Mahmood, A. A. Alsanad, and A. Gumaei, “A systematic study to improve the requirements engineering process in the

- domain of global software development,” *IEEE Access*, vol. 8, pp. 53374–53393, 2020.
- [93] A. Kannenberg and H. Saiedian, “Why software requirements traceability remains a challenge,” *CrossTalk: The Journal of Defense Software Engineering*, vol. 22, no. 5, pp. 14–19, 2009.
- [94] K. Pohl, *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010.
- [95] G. Regan, F. McCaffery, K. McDaid, and D. Flood, “The barriers to traceability and their potential solutions: Towards a reference framework,” in *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, pp. 319–322, IEEE, 2012.
- [96] P. Mäder and A. Egyed, “Do developers benefit from requirements traceability when evolving and maintaining a software system?,” *Empirical Software Engineering*, vol. 20, pp. 413–441, 2015.
- [97] F. Tian, T. Wang, P. Liang, C. Wang, A. A. Khan, and M. A. Babar, “The impact of traceability on software maintenance and evolution: A mapping study,” *Journal of Software: Evolution and Process*, vol. 33, no. 10, p. e2374, 2021.
- [98] G. Spanoudakis and A. Zisman, “Software traceability: a roadmap,” in *Handbook of software engineering and knowledge engineering: vol 3: recent advances*, pp. 395–428, World Scientific, 2005.
- [99] S. K. Sundaram, J. H. Hayes, A. Dekhtyar, and E. A. Holbrook, “Assessing traceability of software engineering artifacts,” *Requirements engineering*, vol. 15, pp. 313–335, 2010.
- [100] S. Maro, A. Anjorin, R. Wohlrab, and J.-P. Steghöfer, “Traceability maintenance: factors and guidelines,” in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pp. 414–425, 2016.
- [101] J. Cleland-Huang, “Toward improved traceability of non-functional requirements,” in *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pp. 14–19, 2005.
- [102] S. Nair, J. L. De La Vara, and S. Sen, “A review of traceability research at the requirements engineering conference re@ 21,” in *2013 21st IEEE International Requirements Engineering Conference (RE)*, pp. 222–229, IEEE, 2013.
- [103] S. Demi, M. Sanchez-Gordon, and R. Colomo-Palacios, “What have we learnt from the challenges of (semi-) automated requirements traceability? a discussion on blockchain applicability,” *IET Software*, vol. 15, no. 6, pp. 391–411, 2021.

- [104] S. Porru, A. Pinna, M. Marchesi, and R. Tonelli, “Blockchain-oriented software engineering: challenges and new directions,” in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pp. 169–171, IEEE, 2017.
- [105] G. Destefanis, M. Marchesi, M. Ortu, R. Tonelli, A. Bracciali, and R. Hierons, “Smart contracts vulnerabilities: a call for blockchain software engineering?,” in *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pp. 19–25, IEEE, 2018.
- [106] C. C. Agbo, Q. H. Mahmoud, and J. M. Eklund, “Blockchain technology in healthcare: a systematic review,” in *Healthcare*, vol. 7, p. 56, MDPI, 2019.
- [107] S. E. Chang and Y. Chen, “When blockchain meets supply chain: A systematic literature review on current development and potential applications,” *Ieee Access*, vol. 8, pp. 62478–62494, 2020.
- [108] M. Marchesi, “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)*, pp. 1–1, IEEE, 2018.
- [109] R. Colomo-Palacios, “Cross fertilization in software engineering,” in *Systems, Software and Services Process Improvement: 27th European Conference, EuroSPI 2020, Düsseldorf, Germany, September 9–11, 2020, Proceedings 27*, pp. 3–13, Springer, 2020.
- [110] S. Demi, R. Colomo-Palacios, and M. Sanchez-Gordon, “Software engineering applications enabled by blockchain technology: A systematic mapping study,” *Applied sciences*, vol. 11, no. 7, p. 2960, 2021.
- [111] M. Beller and J. Hejderup, “Blockchain-based software engineering,” in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pp. 53–56, IEEE, 2019.
- [112] M. Król, S. Reñé, O. Ascigil, and I. Psaras, “Chainsoft: collaborative software development using smart contracts,” in *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pp. 1–6, 2018.
- [113] M. Yilmaz, S. Tasel, E. Tuzun, U. Gulec, R. V. O’Connor, and P. M. Clarke, “Applying blockchain to improve the integrity of the software development process,” in *Systems, Software and Services Process Improvement: 26th European Conference, EuroSPI 2019, Edinburgh, UK, September 18–20, 2019, Proceedings 26*, pp. 260–271, Springer, 2019.
- [114] V. Lenarduzzi, M. I. Lunesu, M. Marchesi, and R. Tonelli, “Blockchain applications for agile methodologies,” in *Proceedings of the 19th International Conference on Agile Software Development: Companion*, pp. 1–3, 2018.

-
- [115] M. S. Farooq, Z. Kalim, J. N. Qureshi, S. Rasheed, and A. Abid, “A blockchain-based framework for distributed agile software development,” *IEEE Access*, vol. 10, pp. 17977–17995, 2022.
- [116] J. C. B. RP, K. Singi, V. Kaulgud, K. K. Phokela, and S. Podder, “Framework for trustworthy software development,” in *2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*, pp. 45–48, IEEE, 2019.
- [117] K. Singi, V. Kaulgud, R. J. C. Bose, and S. Podder, “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, IEEE, 2019.
- [118] R. J. C. Bose, K. K. Phokela, V. Kaulgud, and S. Podder, “Blinker: A blockchain-enabled framework for software provenance,” in *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 1–8, IEEE, 2019.
- [119] K. Singi, V. Kaulgud, R. J. C. Bose, and S. Podder, “Shift-software identity framework for global software delivery,” in *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, pp. 122–128, IEEE, 2019.
- [120] K. Singi, V. Kaulgud, R. J. C. Bose, S. G. Choudhury, S. Podder, and A. P. Burden, “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, IEEE, 2020.
- [121] J. W. Creswell and V. L. P. Clark, *Designing and conducting mixed methods research*. Sage publications, 2017.
- [122] B. Kitchenham, “Guidelines for performing systematic literature reviews in software engineering,” 2007.
- [123] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” vol. 2, 01 2007.
- [124] S. Jalali and C. Wohlin, “Systematic literature studies: database searches vs. backward snowballing,” in *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, pp. 29–38, 2012.
- [125] M. Kuhrmann, D. M. Fernández, and M. Daneva, “On the pragmatic design of literature studies in software engineering: an experience-based guideline,” *Empirical software engineering*, vol. 22, pp. 2852–2891, 2017.

- [126] S. Demi, M. Sánchez-Gordón, and R. Colomo-Palacios, “What have we learned from the challenges of (semi-) automated requirements traceability? A discussion on blockchain applicability,” 1 2021. Accessed: 2021-01-29.
- [127] S. Demi, R. Colomo-Palacios, and M. Sánchez-Gordón, “Software Engineering Applications enabled by Blockchain Technology: A Systematic Mapping Study,” 2021. Accessed: 2021-02-09.
- [128] B. Oates, *Researching Information Systems and Computing*. Researching Information Systems and Computing, SAGE Publications, 2006.
- [129] J. Recker, *Scientific research in information systems: a beginner’s guide*. Springer, 2013.
- [130] S. Demi, M. Sánchez-Gordón, and M. Kristiansen, “Blockchain for Requirements Traceability: A Qualitative Approach,” 2022. Accessed: 2022-06-10.
- [131] S. Demi, M. Sánchez-Gordón, and R. Colomo-Palacios, “Trustworthy and Collaborative Requirements Traceability: Validation of a Blockchain-enabled Framework,” 2022. Accessed: 2022-09-04.
- [132] V. Lenarduzzi, O. Dieste, D. Fucci, and S. Vegas, “Towards a methodology for participant selection in software engineering experiments: A vision of the future,” in *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 1–6, 2021.
- [133] B. Amir and P. Ralph, “There is no random sampling in software engineering research,” in *Proceedings of the 40th international conference on software engineering: companion proceedings*, pp. 344–345, 2018.
- [134] S. Baltes and P. Ralph, “Sampling in software engineering research: A critical review and guidelines,” *Empirical Software Engineering*, vol. 27, no. 4, p. 94, 2022.
- [135] R. Fehring, “Methods to validate nursing diagnoses,” *Heart & Lung*, 1987.
- [136] E. Herranz, R. C. Palacios, A. de Amescua Seco, and M.-L. Sánchez-Gordón, “Towards a gamification framework for software process improvement initiatives: Construction and validation.,” *Journal of Universal Computer Science*, vol. 22, no. 12, pp. 1509–1532, 2016.
- [137] B. Glaser and A. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Observations (Chicago, Ill.), Aldine Transaction, 1967.
- [138] C. Urquhart and W. Fernández, “Using grounded theory method in information systems: The researcher as blank slate and other myths,” *Journal of Information Technology*, vol. 28, no. 3, pp. 224–236, 2013.

-
- [139] K.-J. Stol, P. Ralph, and B. Fitzgerald, “Grounded theory in software engineering research: a critical review and guidelines,” in *Proceedings of the 38th International conference on software engineering*, pp. 120–131, 2016.
- [140] M. Sánchez-Gordón and R. Colomo-Palacios, “Taking the emotional pulse of software engineering—a systematic literature review of empirical studies,” *Information and Software Technology*, vol. 115, pp. 23–43, 2019.
- [141] E. Winter, S. Forshaw, L. Hunt, and M. A. Ferrario, “Advancing the study of human values in software engineering,” in *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 19–26, IEEE, 2019.
- [142] R. Hoda, J. Noble, and S. Marshall, “Developing a grounded theory to explain the practices of self-organizing agile teams,” *Empirical Software Engineering*, vol. 17, pp. 609–639, 2012.
- [143] M. Waterman, J. Noble, and G. Allan, “How much up-front? a grounded theory of agile architecture,” in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, pp. 347–357, IEEE, 2015.
- [144] T. J. Gandomani and M. Z. Nafchi, “An empirically-developed framework for agile transition and adoption: A grounded theory approach,” *Journal of Systems and Software*, vol. 107, pp. 204–219, 2015.
- [145] J. Corbin and A. Strauss, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 2014.
- [146] A. Strauss, J. Corbin, and J. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 1998.
- [147] Y. Chun Tie, M. Birks, and K. Francis, “Grounded theory research: A design framework for novice researchers,” *SAGE open medicine*, vol. 7, p. 2050312118822927, 2019.
- [148] K. Charmaz, *Constructing grounded theory: A practical guide through qualitative analysis*. sage, 2006.
- [149] D. Shin and M. Ibarhine, “The socio-technical assemblages of blockchain system: How blockchains are framed and how the framing reflects societal contexts,” *Digital Policy, Regulation and Governance*, vol. 22, no. 3, pp. 245–263, 2020.
- [150] M. Wiesche, M. C. Jurisch, P. W. Yetton, and H. Krcmar, “Grounded theory methodology in information systems research,” *MIS quarterly*, vol. 41, no. 3, pp. 685–A9, 2017.

Bibliography

- [151] J. Frizzo-Barker, P. A. Chow-White, P. R. Adams, J. Mentanko, D. Ha, and S. Green, "Blockchain as a disruptive technology for business: A systematic review," *International Journal of Information Management*, vol. 51, p. 102029, 2020.
- [152] B. Glaser, *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory*. Advances in the methodology of grounded theory, Sociology Press, 1978.
- [153] K. Krippendorff, *Content analysis: An introduction to its methodology*. Sage publications, 2018.
- [154] J. Coughlan, M. Lycett, and R. D. Macredie, "Communication issues in requirements elicitation: a content analysis of stakeholder experiences," *Information and Software Technology*, vol. 45, no. 8, pp. 525–537, 2003.
- [155] P. Ralph and P. Kelly, "The dimensions of software engineering success," in *Proceedings of the 36th International Conference on Software Engineering*, pp. 24–35, 2014.
- [156] Z. Kurtanović and W. Maalej, "On user rationale in software engineering," *Requirements Engineering*, vol. 23, pp. 357–379, 2018.
- [157] H.-F. Hsieh and S. E. Shannon, "Three approaches to qualitative content analysis," *Qualitative health research*, vol. 15, no. 9, pp. 1277–1288, 2005.
- [158] T. G. Harwood and T. Garry, "An overview of content analysis," *The marketing review*, vol. 3, no. 4, pp. 479–498, 2003.
- [159] U. H. Graneheim and B. Lundman, "Qualitative content analysis in nursing research: concepts, procedures and measures to achieve trustworthiness," *Nurse education today*, vol. 24, no. 2, pp. 105–112, 2004.
- [160] M. Bengtsson, "How to plan and perform a qualitative study using content analysis," *NursingPlus open*, vol. 2, pp. 8–14, 2016.
- [161] D. F. Polit and C. T. Beck, *Essentials of nursing research: Methods, appraisal, and utilization*, vol. 6. Lippincott Williams & Wilkins, 2006.
- [162] T. D. Cook and C. D.T, "Quasi-experimentation: Design and analysis issues for field settings," *Boston Houghtori Mitfliri*, 1979.
- [163] H. T. Reis and C. M. Judd, *Handbook of research methods in social and personality psychology*. Cambridge University Press, 2000.
- [164] M. D. Myers, "Qualitative research in business and management," *Qualitative research in business and management*, pp. 1–364, 2019.
- [165] M. Annells, "Grounded theory method: Philosophical perspectives, paradigm of inquiry, and postmodernism," *Qualitative health research*, vol. 6, no. 3, pp. 379–393, 1996.

-
- [166] R. B. Johnson, "Examining the validity structure of qualitative research," *Education*, vol. 118, no. 2, pp. 282–292, 1997.
- [167] M. El Hussein, S. Hirst, V. Salyers, and J. Osuji, "Using grounded theory as a method of inquiry: Advantages and disadvantages.," *Qualitative Report*, vol. 19, no. 27, 2014.
- [168] C. Boyce and P. Neale, *Conducting in-depth interviews: A guide for designing and conducting in-depth interviews for evaluation input*, vol. 2. Pathfinder international Watertown, MA, 2006.
- [169] J. M. Morse, "Determining sample size," 2000.
- [170] S. N. Goodman, D. Fanelli, and J. P. Ioannidis, "What does research reproducibility mean?," *Science translational medicine*, vol. 8, no. 341, pp. 341ps12–341ps12, 2016.
- [171] S. Demi, "Blockchain-oriented requirements engineering: A framework," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pp. 428–433, IEEE, 2020.
- [172] N. Kshetri and J. Voas, "Blockchain-enabled e-voting," *Ieee Software*, vol. 35, no. 4, pp. 95–99, 2018.
- [173] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and software technology*, vol. 64, pp. 1–18, 2015.
- [174] S. Demi, M. Sánchez-Gordón, and M. Kristiansen, "Blockchain for requirements traceability: A qualitative approach," *Journal of Software: Evolution and Process*, p. e2493, 2022.
- [175] K. A. Schulz, O. J. Gstrein, and A. J. Zwitter, "Exploring the governance and implementation of sustainable development initiatives through blockchain technology," *Futures*, vol. 122, p. 102611, 2020.
- [176] L. F. Capretz, "Bringing the human factor to software engineering," *IEEE software*, vol. 31, no. 2, pp. 104–104, 2014.
- [177] B. Craggs and A. Rashid, "Trust beyond computation alone: Human aspects of trust in blockchain technologies," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, pp. 21–30, IEEE, 2019.
- [178] K. Werbach, *The blockchain and the new architecture of trust*. Mit Press, 2018.
- [179] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, "Blockchain-enabled smart contracts: architecture, applications, and future trends," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266–2277, 2019.

- [180] G. D'mello and H. González-Vélez, "Distributed software dependency management using blockchain," in *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 132–139, IEEE, 2019.
- [181] S. S. Yau and J. S. Patel, "A blockchain-based testing approach for collaborative software development," in *2020 IEEE International Conference on Blockchain (Blockchain)*, pp. 98–105, IEEE, 2020.
- [182] P. T. Duy, D. T. T. Hien, D. H. Hien, and V.-H. Pham, "A survey on opportunities and challenges of blockchain technology adoption for revolutionary innovation," in *Proceedings of the 9th International Symposium on Information and Communication Technology*, pp. 200–207, 2018.
- [183] S. Flovik, R. A. Moudnib, and P. Vassilakopoulou, "Determinants of blockchain technology introduction in organizations: an empirical study among experienced practitioners," *Procedia Computer Science*, vol. 181, pp. 664–670, 2021.
- [184] R. Beck and C. Müller-Bloch, "Blockchain as radical innovation: a framework for engaging with distributed ledgers as incumbent organization," 2017.
- [185] P. D. Dozier and T. A. Montgomery, "Banking on blockchain: An evaluation of innovation decision making," *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1129–1141, 2019.
- [186] N. K. Ostern, F. Holotiuk, and J. Moormann, "Organizations' approaches to blockchain: A critical realist perspective," *Information & Management*, vol. 59, no. 7, p. 103552, 2022.
- [187] S. Farshidi, S. Jansen, S. España, and J. Verkleij, "Decision support for blockchain platform selection: Three industry case studies," *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1109–1128, 2020.
- [188] M. Fowler, J. Highsmith, *et al.*, "The agile manifesto," *Software development*, vol. 9, no. 8, pp. 28–35, 2001.
- [189] J. Cleland-Huang, "Traceability in agile projects," *Software and Systems Traceability*, pp. 265–275, 2012.
- [190] M. Janssen, V. Weerakkody, E. Ismagilova, U. Sivarajah, and Z. Irani, "A framework for analysing blockchain technology adoption: Integrating institutional, market and technical factors," *International Journal of Information Management*, vol. 50, pp. 302–309, 2020.

-
- [191] M. S. Farooq, M. Ahmed, and M. Emran, “A survey on blockchain acquainted software requirements engineering: Model, opportunities, challenges, and future directions,” *IEEE Access*, vol. 10, pp. 48193–48228, 2022.
- [192] J. Marjanović, N. Dalčeković, and G. Sladić, “Blockchain-based model for tracking compliance with security requirements,” *Computer Science and Information Systems*, no. 00, pp. 60–60, 2022.
- [193] D. Damian and J. Chisan, “An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management,” *IEEE Transactions on Software Engineering*, vol. 32, no. 7, pp. 433–453, 2006.
- [194] S. L. Pfleeger and B. A. Kitchenham, “Principles of survey research: part 1: turning lemons into lemonade,” *ACM SIGSOFT Software Engineering Notes*, vol. 26, no. 6, pp. 16–18, 2001.
- [195] H. Schwarz, J. Ebert, and A. Winter, “Graph-based traceability: a comprehensive approach,” *Software & Systems Modeling*, vol. 9, pp. 473–492, 2010.
- [196] R. Elamin and R. Osman, “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, IEEE, 2018.
- [197] L. Goasduff, “Gartner top 10 trends in data and analytics for 2020.” <https://www.gartner.com/smarterwithgartner/gartner-top-10-trends-in-data-and-analytics-for-2020>, 2020. Accessed: 2023-01-31.
- [198] Gartner, “Graph database management solution market guide.” <https://info.cambridgesemantics.com/graph-database-management-solution-market-guide-gartner>. Accessed: 2023-01-31.

Part II: Research Papers



PAPER I

S. Demi "*Blockchain-oriented Requirements Engineering: A Framework,*" in 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 428–433, IEEE, 2020.



PAPER II

S. Demi, M. Sanchez-Gordon, and R. Colomo-Palacios, "*What have we learnt from the challenges of (semi-) automated requirements traceability? A discussion on blockchain applicability,*" IET Software, vol. 15, no. 6, pp. 391–411, 2021.

REVIEW

What have we learnt from the challenges of (semi-) automated requirements traceability? A discussion on blockchain applicability

Selina Demi | Mary Sanchez-Gordon  | Ricardo Colomo-Palacios 

Faculty of Computer Sciences, Ostfold University College, Halden, Norway

Correspondence

Ricardo Colomo-Palacios, Faculty of Computer Sciences, Ostfold University College, Halden, 1757 Norway.

Email: ricardo.colomo-palacios@hiof.no

Funding information

This work is part of a PhD project, funded by Ostfold University College

Abstract

Over the last 3 decades, researchers have attempted to shed light into the requirements traceability problem by introducing tracing tools, techniques, and methods with the vision of achieving ubiquitous traceability. Despite the technological advances, requirements traceability remains problematic for researchers and practitioners. This study aims to identify and investigate the main challenges in implementing (semi-)automated requirements traceability, as reported in the recent literature. A systematic literature review was carried out based on the guidelines for systematic literature reviews in software engineering, proposed by Kitchenham. We retrieved 4530 studies by searching five major bibliographic databases and selected 70 primary studies. These studies were analysed and classified according to the challenges they present and/or address. Twenty-one challenges were identified and were classified into five categories. Findings reveal that the most frequent challenges are technological challenges, in particular, low accuracy of traceability recovery methods. Findings also suggest that future research efforts should be devoted to the human facet of tracing, to explore traceability practices in organisational settings, and to develop traceability approaches that support agile and DevOps practices. Finally, it is recommended that researchers leverage blockchain technology as a suitable technical solution to ensure the trustworthiness of traceability information in interorganisational software projects.

1 | INTRODUCTION

The concept of requirements traceability (RT) was introduced more than 30 years ago by researchers and practitioners [1]. Traceability has evolved from just tracing requirements to implementation and test artefacts [2], to playing a significant role in various software and systems' engineering activities, such as change and defect management [2, 3], project management [3], validation and verification [4], software maintenance [5], and impact analysis [6]. Traceability is particularly important in safety-critical systems [7–9] as it ensures safety, which is crucial for systems whose failure may result in the loss of life, loss or misuse of sensitive information, and major financial loss.

According to Gotel and Finkelstein [10], RT refers to 'the ability to describe and follow the life of a requirement, in both a

forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)'. The typical traceability process model is described by Gotel et al. [11] and entails the creation, maintenance and use of trace links within the scope of a defined traceability strategy. The manual creation of trace links poses the risk of inconsistencies, particularly in complex software projects with a variety of artefacts and relations among them. To reduce the burden of manual tracing tasks, which are time consuming and tedious, automation is of major importance [12]. In fact, enhancing the automatic degree of tracing activities was identified as the second most important research topic in a recent systematic literature review (SLR) on RT technologies [13]. In this regard, information retrieval techniques have been widely

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2021 The Authors. *IET Software* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

shown to support the (semi-)automated creation of trace links [14]. However, these textual-based techniques tend to generate traceability matrices with high recall and low precision [15]. Due to the high amount of false positives generated by these techniques, human analysts are required to vet candidate trace links and make the final decision on whether to accept or reject these trace links [16].

Despite the vast body of knowledge that exists on traceability, traceability practices are far from being mature [17]. Over the years, RT in general and (semi-)automated RT in particular have been widely identified as problematic by the industry [18–23]. According to the NaPIRE (Naming the Pain in Requirements Engineering) survey, 32.4% (158 out of 488) of the respondents positioned ‘missing traceability’ in the top 15 requirements engineering (RE) challenges [24]. This is a concerning value, given the consequences that can reverberate throughout the software development process and product due to the lack of traceability, such as less maintainable software or defects due to inconsistencies [17].

Traceability researchers strived to explore requirements traceability challenges, which led to a lack of implementation or an implementation of RT in a haphazard manner [21, 25–27]. The foundational work in this dimension is laid by Gotel and Finkelstein [10]. These authors attribute the poor requirements traceability practices to the lack of a common definition and the perception that traceability is expected to address conflicting problems in the context of different users, projects and tasks. Further, the Centre of Excellence for Software Traceability (CoEST) published a technical report in 2011 [28], in which traceability researchers and practitioners within the CoEST presented their vision for traceability and identified eight challenges that need to be addressed to achieve the vision. The last challenge entails making traceability ubiquitous, that is, traceability that is built into the system or software engineering process. This challenge has been named *the grand challenge of traceability*, as it requires progress with the seven remaining challenges: purposed, cost-effective, configurable, trusted, scalable, portable, and valued [28].

Our study aims to contribute to this body of knowledge by identifying and classifying challenges of (semi-)automated RT, as reported in the recent literature. To achieve this aim, we carried out a systematic literature review and reviewed 70 primary studies. To the best of our knowledge, the last similar literature review was conducted in 2009 by Winkler and Pilgrim [17]. Therefore, there is a need for an updated and comprehensive study that intends to provide a holistic view of (semi-)automated RT challenges (for related works, see Section 2).

The contributions of this study are as follows: (i) the identification of 21 domain-agnostic challenges and their classification into five categories, (ii) the proposal of blockchain as a suitable solution to ensure reliability and availability of traceability across organisational boundaries, (iii) the identification of research gaps and the call for further research efforts in the following dimensions: *distributed traceability*, *human factors*, *traceability approaches in agile and continuous software engineering*, and *more exploratory studies* in order to enhance the comprehension of traceability practices.

The remainder of the study is organised as follows: We present the related works in Section 2 and the research methodology in Section 3. In Section 4, we introduce the results in terms of the primary studies’ overview and the identification of the challenges. These results are discussed in relation to the related work in Section 5, along with directions for future research and validity threats. Finally, we conclude the study in Section 6.

2 | RELATED WORK

A summary of the related works is provided in Table 1 and it indicates that the last comprehensive literature review focussed on RT challenges was conducted in 2009 and published in 2010. In what follows, we present the related works:

Ramesh [29] conducted surveys with participants in 26 organisations and classified them into low-end and high-end users of traceability, according to their underlying motivation to implement traceability. While low-end users perceive traceability as a mandate, high-end users perceive traceability as an important quality attribute of system and software engineering. This study also presents factors influencing the practice of requirements traceability and categorises them into environmental, organisational and system development contexts. However, the emergence of distributed development and agile paradigms introduces the need for a new study in this dimension.

Blaauboer et al. [18] conducted a case study in order to identify factors influencing the decision to use requirements traceability. Given the management perspective of this study, the following factors were identified: organisation awareness, customer awareness, return on investment, stakeholder preferences, and process flow. The authors emphasised the lack of awareness among software project leaders regarding traceability. Conversely, the scope of our study is not limited to the management perspective.

Kannenberg and Saiedian [30] identified the following challenges of implementing RT: cost, change management, different stakeholders’ viewpoints, organisational problems, and poor tool support. They observed that many organisations struggle to comprehend the importance and benefits of traceability and suggested organisational changes and better tool support in order to reap these benefits. Winkler and Pilgrim [17] carried out an extensive literature review that was conducted in 2009 and published in 2010. The goal of this study was to explore traceability in requirements engineering and model-driven development. These authors pointed out natural, technical, economical, and social factors that hinder the implementation of traceability practices in the industry. Additionally, they recommended further research on improving the human factor in traceability, enabling distributed traceability and providing support for practical problems in the industry. Our study follows a similar approach but provides an updated view of (semi-)automated RT challenges, as reported by the recent primary studies.

Torkar et al. [26] identified requirements traceability tools, techniques and challenges by reviewing studies published

TABLE 1 Related works

Study reference	Author(s)	Publication year	Focus	Research method	# Studies/Organisations	Period of SLR
[29]	Ramesh	1998	Influencing factors on the use and adoption of RT	Evaluation research	26 organisations	-
[18]	Blaauboer et al.	2007	Influencing factors on the decision to use RT	Evaluation research	1 organisation	-
[30]	Kannenber and Saiedian	2009	Challenges of RT	Review	-	-
[17]	Winkler and Pilgrim	2010	Traceability in RE and model-driven development and their challenges	Review	-	-
[26]	Torkar et al.	2012	Factors hindering the implementation of RT	SLR and evaluation research	52 primary studies + 2 organisations	1997–2007
[2]	Regan et al.	2012	Barriers faced when implementing RT	Review	8 primary studies	2005–2011
[31]	Nair et al.	2013	The evolution of the RT research in the RE conference	SLR	70 primary studies	1993–2012
[4]	Mustafa and Labiche	2017	Modelling traceability among heterogeneous artefacts	SLR	330 primary studies	2000–2016
[13]	Wang et al.	2018	RT technologies	SLR	114 primary studies	2000–2016
[27]	Maro et al.	2018	Traceability challenges on the automotive domain	Tertiary study + multi-vocal literature review + evaluation research	24 secondary studies + 245 sources + 1 organisation	2007–2017

Abbreviations: RE, requirements engineering; RT, requirements technology; SLR, systematic literature review.

within the period 1997–2007. According to these authors, cost is the main factor that hinders the implementation of adequate traceability practices. Regan et al. [2] reported 11 traceability challenges that were categorised into management, social, and technical issues. Some of the reported challenges are cost, lack of guidance, political issues, and tool issues. Although these authors advocate the relevance of the challenges in both general and safety-critical domains, they acknowledge that the need to ensure accountability in safety-critical domains poses further complexities to the implementation of automated traceability.

Nair et al. [31] explored the evolution of requirements traceability research by reviewing studies published in the requirements engineering conference (RE) within the period 1993–2012. This review addressed various aspects of requirements traceability, including their challenges. The authors report an increasing interest in automated traceability and suggest traceability visualisation as an area for further research. On the contrary, our systematic literature review (SLR) covers recently published studies, which are not constrained to a specific conference or journal.

Mustafa and Labiche [4] reviewed studies that focussed on traceability in heterogeneous systems. The authors report a minimal research effort on modelling traceability among heterogeneous artefacts and call for more research in this dimension. Our study takes into account requirements traceability among heterogeneous artefacts, albeit in a more general

approach. Furthermore, Wang et al. [13] carried out a systematic literature review to identify RT technologies and their respective challenges. Their findings indicated the following challenges: automated, trustworthy, lightweight, scalable, dynamic, tracing non-functional requirements, value-perceptible, cost-effective, coordinated, and expressible. Our study, however, takes a different approach by explicitly identifying challenges throughout the tracing process, independent of specific technologies. For instance, our study takes into account the human factor in traceability, which is outside the scope of Wang et al.'s review [13].

Recently, Maro et al. [27] conducted a tertiary literature review, a multi-vocal literature review and a case study in the automotive domain. They identified 22 traceability challenges and categorised them into 7 groups: human factors, uses of traceability, knowledge of traceability, tool support, organisation and processes, measurement of traceability and exchange within and across organisations. Instead of that, we carry out a systematic literature review that intends to identify and classify domain-independent challenges of (semi-)automated RT.

Table 1 summarises the aforementioned related works and points out the main differences in terms of the publication year, research focus, research method, number of studies/organisations and the period of the SLR. We notice a limited number of studies that focus on investigating the challenges of RT practices, which are referred to as evaluation research according to the requirements engineering paper classification

proposed by Wieringa et al. [32]. Additionally, the number of organisations considered by these four studies is low, except from Ramesh's [29] study. The other set of studies are literature reviews (tertiary, systematic literature reviews or multivocal reviews) that explore RT challenges as reported by the previous literature. The number of studies included in these reviews range from eight [2] to 330 primary studies [4]. Although there are few literature reviews related to this topic, the last one that aims to explore domain-agnostic challenges of implementing (semi-)automated requirements traceability was carried out by Winkler and Pilgrim [17] in 2009. Therefore, as mentioned before, there is a need for an updated comprehensive study in this area.

3 | RESEARCH METHOD

We carried out a systematic literature review that relies on the guidelines for performing systematic reviews in software engineering proposed by Kitchenham [33]. In what follows, the review process is thoroughly explained [33].

3.1 | Planning the review

The review protocol consists of relevant research question(s), search strategy and inclusion/exclusion criteria. We developed the protocol via brainstorming sessions and performed searches separately in two rounds. The feedback from these rounds was used to uncover problems in the initial version of the protocol and to improve its effectiveness.

3.1.1 | Research question

This study is aimed at identifying and classifying the challenges in implementing (semi-)automated RT, as reported by the recent literature. Therefore, we raise the following research question:

RQ: *What are the challenges in implementing (semi-)automated RT that have been reported in literature?*

The research question was designed according to the question structure proposed by Kitchenham [33]. In this regard, we define requirements traceability as the *population* or the subject of the study, (semi-)automation as the *intervention* and challenges as the *outcomes*.

3.1.2 | Search strategy

After setting the scope of this study, we defined the search strategy, search string and online databases, based on our experience with SLRs in software engineering [34–36]. This process was done prudently to ensure the precision of search strings and, consequently, minimise threats, such as inappropriate results, missing relevant studies or an increased overhead [37]. Regarding search strategies, Jalali and Wohlin [38]

compared two main search strategies for systematic literature reviews—database search and backward snowballing—and concluded that there are no significant differences between the conclusions and patterns derived from these approaches. In this regard, we employed database search as the first-step search strategy, since it is the recommended approach in software engineering [38]. Nonetheless, Jalali and Wohlin [38] also pointed out that database searches lead to a lot of noise, meaning a higher number of irrelevant studies than included studies. Therefore, to mitigate the potential risk of overlooking relevant studies, we also applied the backward snowballing technique, complementary to the database search [33]. The reference lists of the studies selected were scanned and went through a three-stage selection process (see Figure 1).

The formulated query aims to identify the wide spectrum of available literature focussed on (semi-)automated requirements traceability. We identified three main terms and their respective forms. The first two terms are related to the subject of the study, which is requirements traceability. We also used the term ‘software’, since we noticed that researchers tend to overlook the term ‘requirements traceability’ and use ‘software traceability’ instead. Moreover, we incorporated the term ‘tracing’, as an alternative form of ‘traceability’. The last group of terms consists of automated and (semi-)automated, which have been used interchangeably in the RT literature. It is noteworthy that we conducted trial searches with other terms, for example, ‘semi-automatic’, ‘semi-automation’, and ‘assisted’; however, we did not identify further relevant studies. We concatenated these three groups of terms using the Boolean operator ‘AND’, and their forms using the Boolean operator ‘OR’. The final search string was *(automated OR semi-automated) AND (requirements OR software) AND (traceability OR tracing)*.

Given that different search engines have different requirements, for instance, some search engines do not allow nesting, we tailored our search string to these requirements. The search string was issued in five major online databases, as suggested by Kuhmann et al. [37].

The search string was executed in each of the databases in two rounds: first, in February 2020 and finally in June 2020. We delimited the time period to 2009–2019 because the last literature review similar to our approach was conducted in 2009 (see Section 2). We also employed the filter ‘Computer Science’, when available. Finally, a set of 4530 studies was retrieved, as shown Table 2.

3.1.3 | Inclusion and exclusion criteria

Given the high amount of studies retrieved from the database search, we followed a rigorous and reproducible selection process by defining the inclusion and exclusion criteria [37] as follows:

A) Inclusion criteria

- The study must be published within the period 2009–2019.

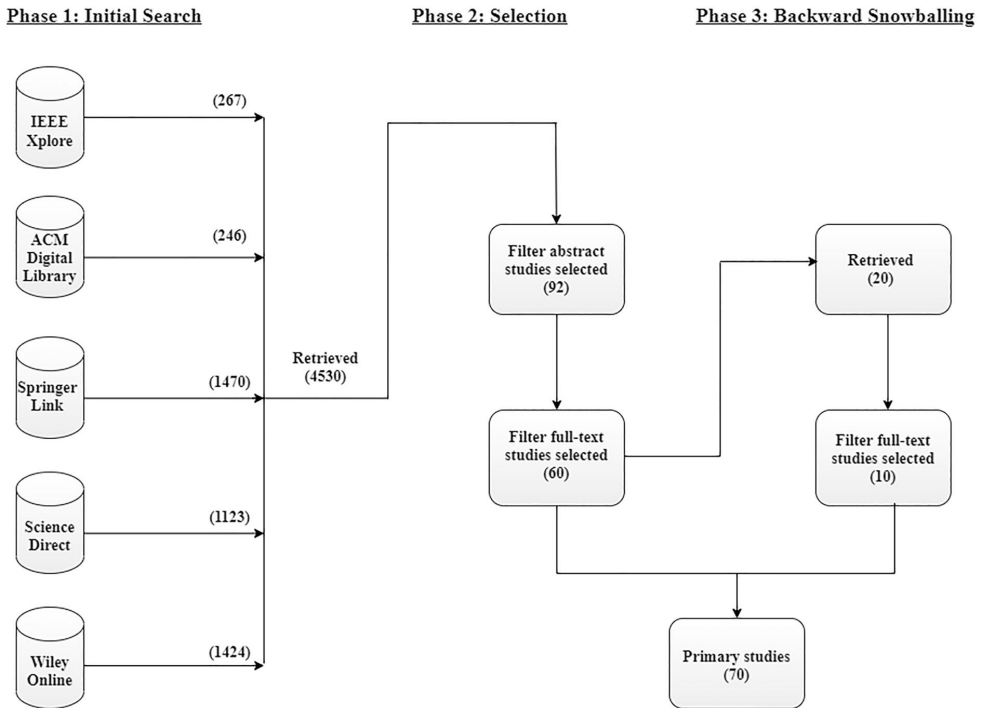


FIGURE 1 Three-stage selection process

TABLE 2 Overview of studies

Database	Initial search	First exclusion	Final exclusion
IEEE Xplore	267	52	36
ACM Digital Library	246	13	8
Springer Link	1470	17	8
Science Direct	1123	7	5
Wiley Online	1424	3	3
Selected studies	4530	92	60
Backward snowballing			10
Primary studies			70

- The study must be available as a full-text article.
 - The study is included if it identifies or addresses at least one challenge of (semi-)automated requirements traceability.
- B) Exclusion criteria
- The study is excluded if it is not written in English.
 - The study that focusses on traceability in domains other than software engineering is excluded.
 - The study is excluded if it focusses on manual traceability.

- If the same article is available in more than one database, the versions available in databases other than the one that provides the article for download are excluded [37].
- If a conference article is followed by a journal article, the conference article is excluded (given that a journal article is a higher-value publication and extends the conference publication) [37].

3.2 | Conducting the review

3.2.1 | Selection of studies

The selection process consisted of three stages, as depicted in Figure 1. First, we retrieved studies from online databases using the search strategy defined in Section 3.1.2. This first stage generated a total of 4530 studies. After removing duplicates, we assessed the title, abstract, and introduction/conclusion (when necessary) of the studies, against the inclusion/exclusion criteria. In this stage, we selected 92 studies and collected their full texts using the reference manager, Zotero. Further, we independently read the full texts and made individual decisions regarding the final selected studies, based on their quality criteria.

ID	Quality question
1	Does the study clearly define the aim?
2	Is the chosen research method appropriate to the research questions?
3	Is the research methodology explained in detail ensuring reliability, internal/external validity, and replicability?
4	Does the study discuss/identify/address any challenges of (semi-)automated requirements traceability by using empirically measured data?
5	Do the conclusions answer the research question(s)?

TABLE 3 Quality questions

Despite some differences in our evaluations, concordance between quality scores was achieved. Disagreements were resolved through discussions. For instance, studies such as [26, 27], which use both the literature review and empirical research methods, were included after discussions because they were evaluated to provide relevant findings. After achieving consensus, we selected 60 final primary studies and excluded the remaining 32 studies (for the list of excluded studies, see [39]). Studies were excluded if they focused on benefits of RT, for example, [5, 25]; if they were short versions of other studies (in such a case we selected the extended version), for example, [40, 41]; if they were secondary studies, for example, [4, 13, 30]. Further, we scanned the reference lists of the selected primary studies and retrieved 20 studies. These studies went through the second stage of the selection process and 10 of them were selected. Finally, a total of 70 studies composed the final set of primary studies (see Appendix).

3.2.2 | Study quality assessment

According to Kitchenham [33], an agreed-upon definition of 'quality of studies' does not exist. Bearing this in mind, we formulated quality questions based on our experience with SLRs in software engineering [34–36] and assessed the selected studies accordingly (see Table 3). We evaluated the studies with the following scores: 0 (does not fulfil the criteria), 0.5 (partially fulfils the criteria), and 1 (fulfils the criteria). A threshold of at least 60% of the maximum score (>3 out of 5) was chosen, to ensure the quality and relevance of the selected studies. Each of the authors computed the quality score of the studies independently. In order to ensure the consistency of the study quality assessment, we computed Krippendorff's alpha (α). The value of this parameter was 78%, which dictates a similarity in the interpretation of data among the co-authors. However, few discrepancies were identified, discussed and resolved through consensus.

3.3 | Data extraction and synthesis

We used the reference manager Zotero to automatically extract data for each study regarding the following attributes: the title of the study, author(s) name(s), date of publication, and the source of publication (journal, conference, symposium, and

workshop). The automatic extraction of this data prevents inconsistencies, which may lead to erroneous analysis and interpretation of the findings. Two of the authors of the study extracted the data independently, by means of an extraction form. The extraction form and the data are available online as archived open data [39]. Other attributes of the data extraction form were inserted manually. For instance, research methods were classified into experiments, surveys and case studies, using the definitions of empirical research methods in software engineering [42]. According to Wohlin et al. [42], these research methods are non-competing; on the contrary, they can be used together in order to enable more informed decisions in software engineering. Therefore, we also investigated combinations of these research methods. In addition, the data extraction form contains the quality scores and challenges addressed by the primary studies. The authors identified themes regarding the challenges that the studies identify and/or address independently. These themes were compared and discussed among the authors. In the case of conflicting themes, disagreements were resolved with the assistance of the third co-author. Finally, the themes were grouped according to their underlying nature into five categories: technological, human factors, organisational, communication and collaboration, and regulatory challenges.

Regarding data synthesis, the previous literature has proposed the following two approaches: descriptive/narrative data synthesis and quantitative data synthesis [43]. We followed the descriptive data synthesis approach by identifying themes based on the data extracted from the selected studies. Additionally, we measured the frequencies of these themes, in order to outline dimensions for future research. It is noteworthy that these frequencies do not indicate the importance of the challenges but provide insights into research gaps.

4 | RESULTS

In the following sections, we provide an overview of the primary studies and findings related to our research question.

4.1 | Overview of primary studies

In this section, we present contextual information about the primary studies. In particular, we outline the distribution of

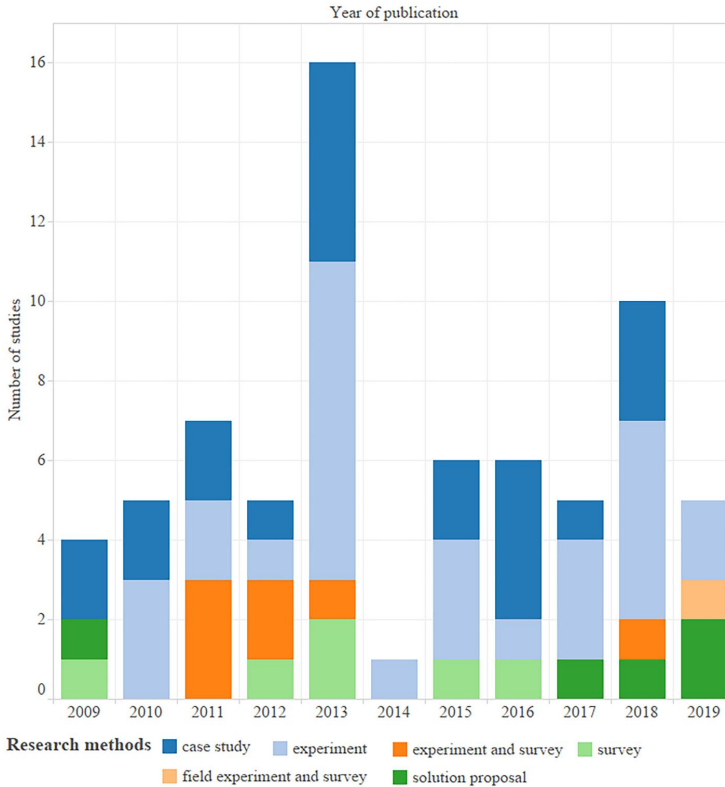


FIGURE 2 Distribution of primary studies based on publication year and research methods

primary studies based on publication year and research methods used, publication type, and quality scores using descriptive representations, such as a simple bar chart, multiple bar chart and a pie chart.

Figure 2 shows the number of primary studies, distributed based on the research method and the year of publication. It is easy to notice the dominance of experiments throughout the majority of the years. For instance, 9 out of 16 studies published in 2013 used experiments as the main research method. Overall, 29 out of 70 studies (41%) adopted only experiments as their research method and 7 out of 70 studies (10%) adopted both surveys and experiments, indicating controlled experiments as the most used research method (36 out of 70 studies). This is not surprising as experiments reduce complexity through the control of all variables other than the ones under investigation, given their reductionist nature [44]. For instance, in [16], the authors aimed to investigate the effect of contextual information in the precision and recall of the final set of trace links and the number of links generated during a specific time period. To reduce complexity, they controlled confounding factors, such as initial precision and recall of trace links. However, controlling such variables may be a limiting factor when it comes to the generalisability of their results and

their applicability in the industry [44]. Therefore, more exploratory studies are needed in realistic environments with practitioners as subjects, in order to explore their feelings, behaviours and attitude regarding traceability practices.

The second observation is related to the number of primary studies distributed over the years. Although a clear trend cannot be identified, we observe a peak of studies in 2013 (16 studies) and a sharp decrease in the number of studies published in 2014. A plausible explanation could be related to the International Workshop on Traceability in Emerging Forms of Software Engineering organised in 2013, as part of the International Conference on Software Engineering (ICSE). This workshop aimed to bring together researchers and practitioners in order to explore the challenges of recovering and maintaining software traceability. The workshop was not organised in 2014, which may explain the low number of studies. In fact, this trend can be also observed in the Requirements Engineering conference. In this regard, we observed the tracks of this conference in 2013 and 2014 with respect to traceability. Our observation revealed three sections dedicated to traceability in 2013: automated traceability, traceability in practice (research track) and traceability in practice (industrial tracks), and only one section in 2014, named traceability.

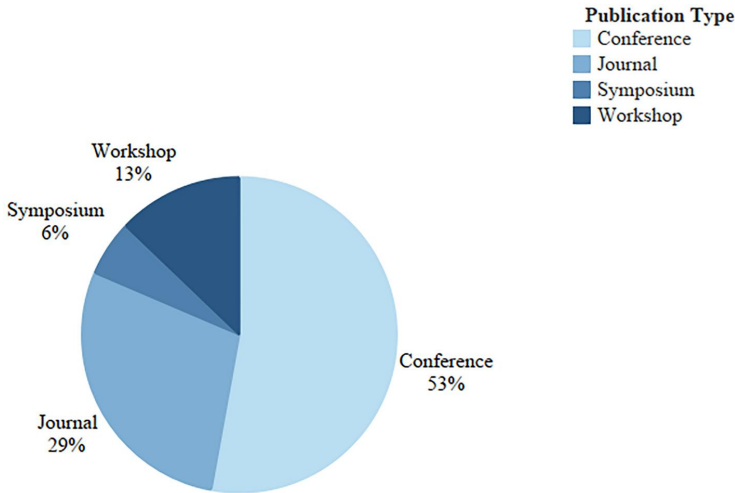


FIGURE 3 Distribution of primary studies based on publication type

The pie chart in Figure 3 shows the relative distribution of primary studies based on publication type. Our findings reveal that the majority of the studies were published in conference proceedings, in total 37 out of 70 (53%). This is not surprising given the conference-driven nature of the software engineering field. Moreover, there were 20 journal studies (29%), 9 workshop studies (13%), and 4 symposium studies (6%). Over 81% of the studies (57 out of 70) were published in journals or conferences, which ensures the quality of the studies. As expected, 32% of the conference studies were published in the Requirements Engineering conference and 44% of the workshop studies in the ICSE International Workshop on Traceability in Emerging Forms of Software Engineering.

The bar chart in Figure 4 depicts the relative frequencies of studies for each of the quality scores. We assigned total scores to the selected studies, by summing up scores of the five quality questions. The results indicate that the selected primary studies scored at least 60% (≥ 3) of the maximum score (5), which is a reasonable threshold. Only 7.14% of the studies scored 3. This low score can be explained by the fact that these studies propose solutions to address challenges of (semi-) automated RT which are neither validated in experimental settings nor evaluated in organisational settings. On the other side, only 10 out of 70 studies reached quality scores of 4.5 and 5. This result can be explained with the fact that most of the studies used experiments as the main research method with students as subjects, which could undermine the validity of the study. Another reason is the lack of discussion regarding threats to validity and reliability that was observed in some of the studies, for instance [7, 45].

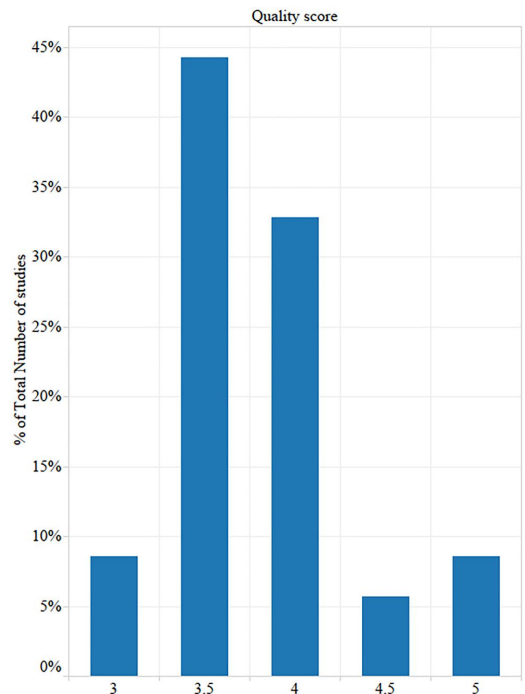


FIGURE 4 Percentage of primary studies for quality score

TABLE 4 Classification of challenges identified from this SLR

Category	Challenges	Literature		
		2009–2014	2015–2019	# Studies
Technological challenges 46 (66%)	Low accuracy of traceability recovery methods	[14, 46–60]	[61–72]	28 (40%)
	Inadequate integration or interoperability among heterogeneous tools	[19, 21, 47, 73–75]	[12, 27, 76–78]	11 (16%)
	Traceability decay	[45, 51, 73, 79]	[69, 76, 80, 81]	8 (11%)
	Lack of change notification and propagation	[82, 83]	[27, 69, 76]	5 (7%)
	Poor presentation and visualisation of trace links	[52, 75, 79, 84]	[27, 85]	6 (9%)
Human factors 18 (26%)	Lack of trust in humans' judgement	[15, 86–91]	[16, 92, 93]	10 (14%)
	Lack of system experience	-	[16]	1 (1%)
	Lack of training	[15, 88, 90]	-	3 (4%)
	Invisible benefits	[84]	[27, 76]	3 (4%)
	Provider-user gap	[79]	[22, 76]	3 (4%)
	Perceived as an overhead	[21, 26, 75]	[27, 76]	5 (7%)
	Organisational challenges 18 (26%)	Lack of organisational strategies and guidelines for traceability	[7, 74, 75, 84, 94]	[27, 95, 96]
Undefined roles and responsibilities for traceability		[75]	[27, 76]	3 (4%)
Project dimensions related challenges		[19, 74]	[65, 97]	4 (6%)
Challenges enabled by the software development approach		[98]	[23, 97, 99–102]	7 (10%)
Communication and collaboration challenges 8 (11%)	Intraorganisational communication challenges	-	[76]	1 (1%)
	Communication challenges in distributed software development	[82]	[27, 69, 76]	4 (6%)
	Interorganisational collaboration challenges	[75, 103]	[12, 76]	4 (6%)
Regulatory challenges 6 (9%)	Implicit traceability requirements in regulations	[7]	-	1 (1%)
	Granularity in requirements for traceability	[7]	-	1 (1%)
	Legal and intellectual property constraints	[75, 103]	[12, 27, 76]	5 (7%)

4.2 | Challenges of (semi-)automated requirements traceability

In this section, we report the challenges of (semi-)automated RT. We identified 21 challenges and organised them into five categories (see Table 4). Due to the relatively large time span, 2009–2019, we decided to split it into two periods—2009–2014 and 2015–2019. We also calculated the number of studies that identify and/or address each of the challenges in order to pinpoint gaps for future research. It is noteworthy that more than one challenge can be identified and/or addressed by a specific study.

4.2.1 | Technological challenges

Our findings indicate that the traceability community has paid more attention towards improving the accuracy of traceability recovery methods than other technological challenges (see Table 4). In the following section, we present the technological challenges identified:

Low accuracy of traceability recovery methods

The most popular methods for generating traceability links are IR methods [13]. These methods have demonstrated low precision (20%–50%), based on experiments conducted in a variety of domains and artefacts [53]. This occurs because IR-based techniques link pairs of artefacts based on their textual similarity, representing only a probability of the relation. We found out a variety of strategies that intend to enhance IR-based techniques. One of the main strategies is relevance feedback, which consists of incorporating incorporation of human judgement to modify initial representations of queries. Shin and Cleland-Huang [57] enable the analyst to directly manipulate individual trace queries by inserting or filtering out the terms. Building on this approach, Dietrich et al. [53] proposed Trace Query Modification (TQM) to expand the benefits of user judgement across multiple queries. TQM uses a set of initial and modified queries to learn transformation rules which are then applied to future trace queries. Panichella et al. [71] introduced the concept of adaptive relevance feedback, which consists of considering information about the software artefacts and already-classified trace links before applying

relevance feedback. While this approach performs relevance feedback for a subset of the links, Wang et al. [68] suggested performing relevance feedback for a subset of terms within each trace link.

Furthermore, it has been reported that IR-based techniques are not able to relate terms with similar meaning, for example, ‘error’ and ‘failure’ [54]. In such a case, the use of a dynamic thesaurus was proposed to deal with synonyms [54]. Another problem that we identified is polysemy, which refers to the same term appearing in different requirements with different meanings. For instance, the term ‘task’ can mean ‘workflow task’ or ‘development task’. To address the polysemy problem, Wang et al. [62] proposed training an artificial neural network to determine whether a term has the same meaning in different requirements. Other enhancing strategies consist of a combination of regular expressions, key phrases and clustering [60], a smoothing filter to filter out recurring terms that bring irrelevant information [58], query expansion by means of web mining [59] and filtering after recovering trace links by means of part-of-speech tagging [72].

However, these approaches fail to leverage the underlying semantic information. In this regard, Li and Cleland-Huang [56] applied a domain-specific ontology combined with generalised ontology to trace artefacts based on their semantics and proved the effectiveness of such a combination. However, building this ontology is time consuming and an inappropriate ontology can actually worsen the quality of trace links by missing relevant relationships or retrieving unrelated artefacts. In this regard, Guo et al. [70] used deep learning to automatically capture the domain knowledge and semantics of artefacts. The proposed approach adopts unsupervised learning techniques to learn word embeddings with respect to the domain and a recurrent neural network to learn the semantic representation of the artefacts.

Inadequate integration or interoperability among heterogeneous tools

The software development lifecycle (SDLC) consists of a variety of tools that generate many artefacts of different formats and specified in different languages. Maro et al. [12] conducted 24 interviews with software development stakeholders and revealed that 14 out of 24 interviewees attributed the difficulty in implementing traceability maintenance to the heterogeneous nature of the tools. Likewise, in a later tertiary study, Maro et al. [27] considered tool integration as technically challenging.

The previous literature has identified three choices regarding traceability among heterogeneous tools [12]. The first and most trivial solution is a holistic tool platform in which everything, including traceability, is fully integrated in the same tool. This solution ensures consistency because the stakeholder who wants to make a change has to first delete or update affected trace links. However, this holistic solution might work in small companies, but it is infeasible in the context of interorganisational collaboration (see Section 4.2.5). A second solution could be a separate traceability management tool. In such a case, elements in the traceability tool need to connect with external models, by means of tool adapters. In this regard,

Asuncion and Taylor [47] explored the integration of tools into an open hypermedia system by using tool-specific adapters that enable an effective means for traceability across heterogeneous tools’ boundaries. Finally, a hybrid solution has been reported, which consists of combining requirements management and traceability management in the same tool. In such a case, there is a need to import models into the traceability management tool. These models can still be changed externally, amplifying, in this way, the inconsistency problem. Therefore, previous research recommends avoiding the hybrid solution [12].

Traceability decay

If trace links are not updated when changes occur, traceability relations deteriorate, that is, some trace links get lost and others represent false relations, leading to the so-called *traceability decay* [73]. This phenomenon is particularly frequent in the case of links between requirements and source code because in most of the cases, developers change the code frequently without updating the links [80]. To address the traceability decay challenge, Mäder and Götzel [73] proposed Trace-Maintainer, a tool that adopts a rule-based approach for the (semi-)automated maintenance of trace links. This tool captures change events while developers perform software development activities using UML (unified modelling language) diagrams. It recognises the software development activity and, consequently, maintains the impacted trace links. A similar recent approach that aims to automate the maintenance of trace links between requirements and source code is the Trace Link Evolver (TLE) [80]. The TLE is based on a set of heuristics coupled with refactoring detection tools and IR algorithms in order to identify pre-defined change cases. However, this approach, in contrast with Mäder and Götzel’s approach [73], does not need a monitored environment.

Lack of change notification and propagation

A change of an artefact causes changes in the connected artefacts and trace links. In the worst-case scenario, the change of an artefact, for example, requirements, affects a chain of artefacts, for example, design, source code, test cases and different departments, for example, mechatronics and software engineering [76]. This indicates the need to notify the affected stakeholders in order to update the related artefacts and trace links. For this purpose, some requirements management tools have a ‘suspect links’ feature, where links are propagated to developers’ local workplace in case of a change and then developers have the responsibility to decide how to update artefacts and traceability relations [27]. However, it is still the user who resolves the change manually and this may lead to inconsistencies.

The distributed development paradigm exacerbates this challenge, as the communication among distributed stakeholders is difficult. To enhance remote teams’ awareness regarding requirements changes, a multi-agent approach was proposed in [69]. When a requirement is updated or deleted, the recommender agent uses the traceability data to determine the impact of the change on artefacts and to identify the creators of these artefacts. Then, the agent sends a message to

the creators of the affected artefacts. When an element is added, the recommender agent identifies interdependencies and the impact of the change. This approach ensures autonomy of agents, as they can operate without direct intervention from users.

Poor presentation and visualisation of trace links

Large-scale projects are characterised by a high number of artefacts and, consequently, a high number of trace links. These trace links are represented by means of lists or mega tables that hinder the comprehension of traceability data and the detection of inconsistencies by interested stakeholders [27, 84]. Recently, Aung et al. [85] proposed a hierarchical trace map visualisation in order to represent relationships among artefacts in an interactive manner. Nodes represent artefacts and are clickable for a filtered view, whereas edges represent the relationship between artefacts. This trace map view supports analysts in system comprehension and change impact analysis.

4.2.2 | Human factors

Due to the aforementioned technological limitations, existing traceability tools yield results that cannot be trusted to certify the process, especially in safety-critical systems. Therefore, humans play a valuable role in the process by validating candidate links generated by tracing tools. Several researchers investigated the performance of human analysts during tracing tasks, while others identified factors that influence such performance and as a result, the accuracy of the tracing process. It is noteworthy that some of the tools stop at the generation phase, and only few of them offer the functionality to validate tracing links [16]. In what follows, we present the challenges related to the human facet of the tracing process. The first three challenges are focussed on humans in the vetting process, while the remaining challenges concern stakeholders interested in creating, maintaining, and using traceability.

Lack of trust in humans' judgement

Cuddeback et al. [86] investigated the performance of human analysts while vetting candidate traceability links, through experiments that involved 26 participants from two universities. The results of this study revealed that 50% of the participants did not improve the accuracy of the links. Instead, most of them decreased the overall accuracy of the candidate links. In line with the results of this study, only a year later, Dekhtyar et al. [87] confirmed that participants failed to recover true traceability links. Their multivariate statistical analysis revealed that high accuracy of candidate traceability links results in low accuracy of final links, due to the fact that humans tend to add bad links or remove good links [90].

We identified few studies that focus on supporting humans in the tracing process. Maro et al. [16] explored the content and context information that could be useful to support human analysts. Their findings suggest that analysts need (i) information from the connected artefacts, for example, who created the artefacts, who modified the artefacts, location

of the artefacts in the system, and other connected artefacts, (ii) information from the traceability information model, and (iii) information from the tracing algorithm. Moreover, Wang et al. [93] investigated whether the use of user-defined keywords, named tagging, supports the analysts effectively. Their results confirmed that tagging significantly improved precision, as analysts can decide whether or not to accept plausible links by looking for keywords. To further improve analysts' performance, these tags can be exchanged among analysts.

Lack of system experience

One may expect a positive correlation between the level of experience in software development and tracing performance. Surprisingly, studies demonstrated a non-significant correlation between experience and performance [87]. Nonetheless, there is evidence that the familiarity or experience with the system is more important than software development experience or tracing experience [16]. In this regard, it has been suggested that the role of vetting tracing links should be allocated based on system experience. For instance, developers should vet links between requirements and code.

Lack of training

It has been reported that analysts spend a significant amount of time on the so-called grey links, that is, neither obvious true links nor obvious false links. This occurs due to the lack of training and direction; in particular, in the way the final traceability matrix (TM) is to be used [15]. Analysts can be trained to use TM characteristics, such as an estimate of the TM size and when they select links to be added into the final TM [15]. The concept of 'educating the user' on how to decide on difficult trace links in an efficient fashion was introduced in 2011 by Cuddeback et al. [90]. The authors elaborated this concept in a later study and proposed adding a training session with a validation task that needs to be passed by the analyst in order to proceed to real traces [88].

Invisible benefits

Although traceability is valuable in many aspects of the software development lifecycle (see Section 1), it has been reported that stakeholders do not perceive traceability benefits [27, 76, 84]. In order to create and maintain quality trace links, interested stakeholders need to be aware of the benefits of traceability [76]. Maro et al. [27] outlined the importance of providing measurements of the direct benefits of traceability; however, they stated that currently there are no such measurements. They proposed quantifying the benefits of traceability by collecting data on the usefulness of traceability links. These data can be collected by monitoring activities affected by traceability and by carrying out surveys with traceability users.

Provider-user gap

Traceability is perceived as an elusive quality attribute of software development because practitioners who create trace links are not the same as practitioners who use these links [22, 76] or depend upon them [79], that is, the so-called provider-user gap. For instance, developers create trace links

from requirements to source code at different granularity levels but another group of stakeholders will eventually use these links, for example, project managers to track progress. In this regard, Wohlrab et al. [76] suggested balancing the effort and benefit for traceability per role.

Perceived as an overhead

Recent studies reported the reluctance of stakeholders to invest in traceability [76] because they perceive traceability as an extra task that disrupts their workflow. The perceived overhead and invisible benefits demotivate stakeholders from prioritising traceability tasks, leading to the creation of wrong or missing trace links. Maro et al. [27] attribute this perception to two main factors: organisational and technical. The organisational factor is related to the provider–user gap and the technical factor is related to poor presentation and visualisation of trace links.

4.2.3 | Organisational challenges

Lack of organisational strategies and guidance for traceability

Regan et al. [7] conducted a traceability assessment in two medical companies and outlined the lack of detailed guidance for implementing traceability as one of the main factors that shapes the perception of traceability as complex and difficult. Practitioners need guidance in taking the decisions about trace paths and traceability usage goals [74], about trace granularity which can lead to either excessively coarse-grained or excessively fine-grained links [75], and in systematically assessing the trace link quality [95]. Rempel et al. [74] revealed that without explicitly defined strategies, practitioners are far away from implementing effective traceability, that is, traceability that supports the development project. The lack of organisational strategies and guidelines for traceability leads to traceability approaches from individual teams in a bottom-up fashion. The creation of links in an ad hoc manner can cause inconsistencies and, consequently, deteriorate the traceability link quality. Mäder et al. [84] recommended defining traceability strategies in the early stages of the project, while Rempel et al. [74] suggested defining a traceability strategy by considering all traceability usage scenarios and goals for each software process task that requires traceability. To provide guidance for systematically assessing trace links' quality, Rempel and Mäder [95] proposed a traceability assessment model (TAM) that identifies for each traceability element an acceptable state and unacceptable deviations from the state.

Undefined roles and responsibilities for traceability

Mäder et al. [75] carried out surveys with 10 practitioners to explore traceability practices. They reported undefined roles for the creation, maintenance, and use of traceability in all the cases under study. Recent studies revealed that the roles and responsibilities for traceability may be defined within an organisational team or discipline but not on a higher organisational level [76]. The interdisciplinary and interorganisational

nature of traceability adds complexity to the coordination of these roles and responsibilities in practice [76]. In inter-organisational software projects, the root of this complexity lies in the divergences between organisations in the following dimensions: different vocabularies, objectives, and development processes [27].

Project dimensions related challenges

The challenges of implementing requirements traceability increase with *project size* and *project complexity*. A large project entails a high number of engineers, a high number of artefacts that need to be linked, and more communication overhead [97]. A complex project consists of a variety of components and interconnections which are difficult to understand, manage or change. The variations of traceability information in terms of format and content complicate the representation of trace links and the understanding of these links by users [65]. Furthermore, Rempel et al. [74] identified *project type* as a factor influencing traceability. They observed that projects in product-oriented companies are characterised by a more homogeneous tool landscape, that is, a holistic tool platform or a highly integrated toolchain, than in service-oriented companies. Thereby, projects in product-oriented companies tend to have less volatile trace paths.

Challenges enabled by software development approach

Espinoza and Garbajosa [98] advocated that existing traceability approaches depend significantly on the characteristics of traditional software development processes. They pointed out two elements to underline why conventional traceability approaches cannot be applied to agile projects: the lack of a specification documentation of the formal requirements in agile approaches and the differences in links semantics. For instance, given the multi-facet nature of user story tests (that act as requirements), links from requirements to user story tests do not have the same meaning as links from requirements to acceptance tests in traditional methodologies. Therefore, the authors outlined the need for customisable traceability models, where trace links' types can be defined according to project needs. In this regard, they proposed a traceability metamodel that supports three features: (i) user-definable traceability, (ii) roles, and (iii) linkage rules.

Furtado and Zisman [102] proposed the Trace++ approach to support the transition from traditional to agile methodologies. This approach tackles four problems of agile projects: (i) absence of metrics to measure the rework per sprint, (ii) lack of understanding of the scope of the project, (iii) lack of documentation about non-functional requirements (NFR), and (iv) absence of management control. Trace++ extends traditional traceability relationships. For instance, to address the lack of documentation about NFR, they add traceability relations between user stories, test scenarios and story acceptance criteria, such as performance and security.

We identified only one study that addresses traceability in DevOps (Development-Operations) environments [99]. DevOps practices foster frequent updates of artefacts with

continuous integration, testing and deployment. Recently, Rubasinghe et al. [99] proposed the SAT Analyser tool with a DevOps extension, to establish traceability between software artefacts in the development and operational level. This approach was evaluated in a case study by means of statistical and network analysis and achieved an accuracy of 71%.

4.2.4 | Communication and collaboration challenges

Intraorganisational communication challenges

Organisations are composed of separate departments and disciplines that often need to communicate and collaborate, for instance electrical, mechanical and software engineers in the automotive domain. However, it has been reported that different disciplines employ different traceability practices [76]. Without effective communication, the links created by one discipline might not be understood by the rest of the practitioners, leading to inconsistent traceability practices throughout the organisation.

Communication challenges in distributed software development

Software development companies are moving towards distributed development teams across multiple remote sites. In this context, communication and coordination challenges arise due to language and cultural differences among distributed stakeholders [69]. In turn, these challenges affect traceability as distributed teams may employ ad hoc and inconsistent traceability practices. A collaborative traceability tool that enables the maintenance of information in a shared space may support communication regarding trace links [76].

Interorganisational collaboration challenges

At a larger scope, challenges are observed even across organisational borders. Rempel et al. [103] attributed the difficulties in implementing traceability across organisational boundaries to the following problem areas: different organisational background of clients and suppliers lead to different technologies and methodologies used, restricted access to artefacts due to organisational boundaries, and conflicting objectives. These authors recommend that practitioners ensure the availability and reliability of traceability, identify and mitigate conflicting objectives and bridge the technological gap between suppliers and clients. A more recent viewpoint was provided by Wohlrab et al. [76]. They conducted multiple case studies to identify collaboration challenges in traceability management and reported very little traceability support for external organisations. These organisations communicate via e-mail and change, delete or insert data manually into their requirements management tools. This may lead to inconsistencies and mistakes, as the update of trace links is not done automatically [76]. The authors attributed this challenge to the heterogeneity of tools used by different organisations and to the fact that suppliers work with a variety of customers without customising their traceability practices [76].

4.2.5 | Regulatory challenges

Implicit traceability requirements in regulations

Requirements traceability is vital for the safe and effective development of safety-critical systems; therefore, it is mandated by domain and country-specific standards and guidelines [8]. Examples of these standards are ASPICE (Automotive Software Performance Improvement and Capability dEtermination) for the automotive industry [27] or GPSV (General Principles of Software Validation) for the healthcare industry [7]. However, it has been reported that the references to traceability are not explicit in regulations [7]. For instance, the European regulation for medical devices, named Medical Device Directive, does not explicitly refer to requirements traceability throughout the software lifecycle. Instead of that, it requires the validation of medical software according to the 'state of the art', which is open to interpretation.

Granularity in requirements for traceability

Standards differ in the level of traceability detail they provide [7]. It has been reported that there are standards that do not mandate traceability throughout the software lifecycle. For instance, IEC 62304 does not require traceability through the design and implementation stages. On the other hand, there are standards such as GPSV that require traceability among the following artefacts: requirements-design-code test, at both the function and module level [7]. The identification of references to traceability within each of the standards with different levels of detail may be time consuming and may complicate the implementation of effective traceability [7].

Legal and intellectual property constraints

Often, there is a need to create traceability links between artefacts of different organisations; for example, in the automotive domain, artefacts are exchanged between organisations due to the OEM (Original Equipment Manufacturer)–supplier relationship [27]. However, in practice, traceability across organisational boundaries is challenging due to legal and intellectual property constraints [12, 27, 76]. For instance, the OEM does not share confidential artefacts that contain intellectual property with suppliers. The restricted access to artefacts complicates the creation of traceability links by suppliers [27].

5 | DISCUSSION

In this section, we discuss our general observations, propose blockchain for requirements traceability and present future research directions. The section concludes with a discussion on the potential threats to validity.

5.1 | General observations

Over the years, researchers have identified a variety of challenges of (semi-)automated requirements traceability. To provide a holistic view of these challenges, we carried out a

systematic literature review. We selected a set of 70 primary studies and observed that most of these studies were published in conference proceedings. A plausible explanation for this could be related to the significant advances in traceability research in the International Requirements Engineering conference over the years. For instance, Nair et al. [31] identified 70 primary studies published in the RE conference within the period 1993–2012. In fact, a closer look at the publication sources of our primary studies reveals that 12 out of 37 conference studies (32%) were published in the proceedings of the RE conference. This is in line with Winkler and Pilgrim's [17] statement that a significant part of traceability research has been conducted by the requirements engineering community.

Furthermore, we observed that over half of the primary studies used experiments (36 out of 70%, 51%) and 5 studies presented solutions without validating them. Consequently, the lack of exploratory approaches, such as case studies and surveys, is one of our findings. In fact, given that RT is an interdisciplinary and complex field, exploratory approaches can contribute by providing empirical evidence of how traceability practices are performed in the industry. Likewise, Niu et al. [31] identified case studies and surveys as the least used empirical methods in traceability studies and they suggested that researchers explore industrial perspectives and experiences. The need for more empirical evidence has been also observed in the wider scope of RE [104].

We observed the following two main dimensions of traceability research regarding the challenges: approaches that address (semi-)automated traceability creation and maintenance challenges and approaches that explore challenges attributed to humans in the tracing process. The findings revealed that the most frequent challenge is the low accuracy of traceability recovery methods. Likewise, Wang et al. [13] concluded that the majority of the selected primary studies focus on improving the trustworthiness degree of trace links. Thus, 40% of our primary studies identified and/or addressed this challenge, whereas human challenges were identified and/or addressed by 25% of the studies. These findings indicate that human factors have not received enough attention in the traceability community. In fact, this is not surprising because human factors in software engineering do not receive the attention they deserve [105].

5.2 | Blockchain applicability

This SLR identified 21 challenges of (semi-) automated requirements traceability which were categorised into five groups: technological, human factors, organisational, communication and coordination challenges, and regulatory challenges (see Table 4). New technologies are called to address these challenges [13]. In this work, we propose the use of blockchain technology for requirements traceability. This proposal is in line with software engineering researchers who advocated the cross fertilisation between hyped technologies such as blockchain and software engineering [106, 107].

Blockchain is a distributed ledger that stores transactions in a chain of blocks [108]. The chain of blocks is created due to the fact that each block contains the hash of the previous block ensuring *immutability*, which can be defined as the inability to tamper with transactions stored on the blockchain [109]. In order for transactions to take place in a *decentralised*, yet *reliable* manner, a variety of core technologies are integrated, such as cryptographic hashes, distributed consensus mechanisms, and digital signatures that rely on asymmetric cryptography [110]. Due to these properties, multiple parties share a single truth via a distributed ledger, which is *verifiable* at any time. Therefore, blockchain can facilitate trusted collaboration and coordination in distributed software development, software provenance, and software integrity assessment [111]. Another important blockchain property is smart contracts, which are self-executing scripts stored on the blockchain to enable reliable transactions and agreement among different trustless parties [112]. Thus, 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 [111].

A recent systematic mapping study carried out to explore the software engineering applications enabled by blockchain technology [111] observed a growing trend of blockchain-oriented software engineering studies during the last 3 years. For instance, Yilmaz et al. [113] proposed the use of blockchain technology to ensure integrity in large-scale agile software development. The authors considered developers as miners who develop code and testers as validators of the code. The incentive mechanism enabled by blockchain technology eliminates the need for project leaders to assign tasks to developers; instead of that developers compete for creating the best code. Other SE researchers have proposed the use of blockchain as a backbone of the SDLC ecosystem [114–116], while Singi et al. [114] presented a blockchain-enabled governance framework to ensure the trustworthiness of the software development process. The framework monitors and captures event data and assesses their adherence to regulations and best practices by means of smart contracts. Finally, Bose et al. [115] introduced a blockchain-enabled framework for reliable software provenance, named Blinker. The framework consists of data ingestion tools for the extraction of data from disparate sources and the transformation of the data in compliance with PROV specifications. Provenance data are validated by means of voting mechanisms or social certifications. The former requires all or a set of participants to approve transactions according to voting policies and the latter relies on participants rating the provenance data based on their perceived benefits. Once consensus is achieved, provenance data are appended to the distributed ledger. Thereby, they cannot be modified or accessed by unauthorised users. To provide insights from provenance information, the framework enables provenance query services that focus on artefacts, agents and processes. Additionally, to enhance comprehension, the authors visualise provenance information through interactive hierarchical graphs.

The aforementioned studies provide inspiration and interesting insights into the potential of using blockchain to address software engineering issues and trigger new promising directions that are not explored in the previous literature on requirement engineering. One of such directions is the use of blockchain for requirements traceability in distributed settings. Traceability information such as artefacts and traceability links that are created by distributed participants can be stored on the blockchain. Due to the inherent properties of blockchain, all authorised stakeholders share a holistic, reliable, and trustworthy traceability knowledge base and may verify its authenticity at any time. This can address interorganisational collaboration challenges with respect to traceability by ensuring the reliability and availability of traceability information [103]. In Section 4.2.4, we outline the restricted access to artefacts due to organisational boundaries as one of the factors that complicates traceability. We argue that the roots of this restriction lie in the lack of trust between parties involved in the development of large-scale software. These trust issues can be mitigated by using blockchain technology.

Another RT challenge that blockchain can address is the reluctance of practitioners to invest in creating quality trace links [76] which is caused by three related factors: the perceived overhead, invisible benefits, and the provider–user gap. In order to motivate participants to participate in traceability tasks, an incentive mechanism can be enabled by smart contracts. Smart contracts can allocate digitised tokens to participants who create quality trace links. Despite the fact that the validation of traceability quality is not trivial and requires manual work [27], this approach can potentially enhance traceability quality which, in turn, may encourage stakeholders to use traceability links to support SDLC tasks. However, the advantages of using blockchain for requirements traceability remain theoretical and further efforts are required to validate them.

5.3 | Future research directions

Our findings suggest the following research directions that are underexplored in the current scientific literature:

5.3.1 | Distributed traceability

The distributed development paradigm and the need for interorganisational collaboration call for distributed traceability management. Creating trace links across organisational boundaries is a challenging task since some of the artefacts are inaccessible due to confidentiality constraints. For instance, OEMs do not share artefacts containing intellectual property that differentiates them in the market [27]. The reduced subset of artefacts that can be accessed by a project partner is not sufficient for achieving complete requirements traceability [103]. Therefore, the project partner has to rely on the traceability information provided by the other partners. Given that mistrust is as a critical issue in interorganisational

projects [117], Rempel et al. [103] required practitioners to ensure reliability and availability of traceability across organisational boundaries. In particular, this requirement can be aligned with the inherent properties of blockchain technology. We perceive blockchain as a promising discipline that can contribute to distributed traceability, as mentioned before in Section 5.2.

5.3.2 | Human factors

We identified limited empirical evidence with respect to human factors in the tracing process. In this regard, more studies are needed in two facets. First, the identification of factors that influence the performance of analysts during the vetting process and the tool support that is needed to assist them, for example, tagging or contextual information about artefacts. Second, it is important to investigate how practitioners can be motivated to invest in trace link quality. We perceive gamification as an appealing area that can contribute to enhancing the motivation and engagement of practitioners in traceability tasks.

5.3.3 | Traceability in agile and continuous software engineering

Traditional RT techniques are infeasible in agile and continuous software engineering environments, due to the absence of requirements specification documents, continuous integration, testing, and delivery. Wang et al. [13] identified only one study that focusses on agile-oriented traceability. In this SLR, we found six additional studies, probably due to the fact that the scope of our study is not limited to RT technologies. However, the popularity that agile and, particularly, DevOps practices are recently gaining indicates that more lightweight traceability approaches are needed to support agile and DevOps practices.

5.3.4 | More exploratory studies

Our findings revealed a low number of exploratory studies that focus on how (semi-)automated RT is performed in industrial environments, what the existing challenges are and the practical needs for traceability. In fact, more empirical evidence could enhance the overall understanding of traceability practices, and henceforth motivate practitioners to implement (semi-)automated requirements traceability.

5.4 | Threats to validity

Although this SLR was conducted with rigour and a reproducibility package is provided [39] to ensure transparency and replicability, few limitations exist. In the following section, we explain the main threats to validity.

Internal validity refers to the degree to which researchers of the study can draw conclusions from causes and effects. A typical threat to internal validity is researchers' biases; for instance, the selection of five databases. Although there is some inevitable subjectivity, this study followed with rigour the guidelines for performing systematic literature reviews in software engineering [43] in order to minimise these biases as much as possible. The guidelines consist of planning and developing a review protocol, developing research questions using the PICO strategy, using backward snowballing in addition to the database search, developing adequate search strings, inclusion/exclusion criteria, and using a data extraction form that includes quality criteria. Moreover, another threat to internal validity could be the maturity of the field. In this regard, given that requirements traceability has been studied for over 3 decades, we believe that the field is mature enough, at least from the research perspective, thus suitable for a systematic review.

External validity refers to the extent to which the findings are generalisable to other contexts. It is worth mentioning that we did not constrain the selection of studies to a specific domain or tracing tool; thereby, the majority of challenges we identified are of general nature. Nonetheless, the interpretation and priority of these challenges could be different in different types of domains. For instance, humans' judgement in the vetting process is very important in safety-critical systems, whereas in general-purpose systems it does not have the same relevance.

Construct validity refers to the extent to which the study measures the construct adequately. To ensure that the selected studies focus on *(semi-)automated requirements traceability*, we used these terms in the search string. Given that the term 'traceability' can be used interchangeably with the term 'tracing', we included both these terms to retrieve as many relevant studies as possible. Additionally, we conducted two searches: first, with the term 'automated' and 'semi-automated' and second, with other terms such as 'assisted'. These two searches retrieved the same set of primary studies. However, we recognise that other terms could be used. To minimise this threat, we performed backward snowballing and identified 10 relevant studies. Although this study cannot ensure completeness, we believe that it includes the most relevant primary studies regarding (semi-)automated requirements traceability.

Conclusion validity refers to reliability, that is, the extent to which results can be relied on to lead to correct conclusions. To ensure reliability, we developed the review protocol via brainstorming sessions, conducted searches in two rounds and cross checked the results of the search process. Moreover, two of the authors assessed the selected studies against quality criteria independently, and the third author assessed the entire process. Disagreements were resolved through long discussions and consensus was achieved for the final set of primary studies. Regarding the data analysis process, we did not conduct it automatically by means of an analysis tool. Manual coding is potentially prone to human errors; however, two of the authors performed the coding process independently, and

their results were assessed by the third author with experience in SLRs in software engineering.

Furthermore, to ensure replicability of our study, we provide our extracted data as an archived package that can be accessed online [39]. In this way, our work can be assessed and/or extended by other researchers. It is noteworthy that there are interdependencies between the categories presented in this study; for instance, change notification and propagation of communication/collaboration challenges in distributed environments or poor visualisation of perceived overhead. However, these interdependencies are outside the scope of this study and are intended for future research. The interpretation of the interdependencies could enhance the understanding of the challenges.

6 | CONCLUSION

We carried out a systematic literature review to shed light on the challenges of implementing (semi-)automated requirements traceability, as reported by the recent literature. A total of 4530 studies were retrieved and 70 of these studies were selected as relevant to this SLR. The objective of this study is to provide a holistic view of (semi-)automated RT challenges in order to encourage further research in this area and to motivate practitioners to implement traceability practices. Our findings indicate experiments as the most frequent research method and a lack of exploratory studies, such as surveys and case studies.

We identified 21 challenges and classified these challenges into the following categories: technological challenges, human factors, organisational challenges, communication and collaboration challenges, and regulatory challenges. The most frequent challenges identified and/or addressed were technological challenges, in particular, the low accuracy of traceability recovery methods. Based on the findings, we also outlined promising dimensions that deserve further research. Blockchain technology was proposed as a suitable technical solution to address distributed traceability. Furthermore, we identified the need to address the human facet of the tracing process in two directions: by exploring how human analysts can be supported during the vetting process and how stakeholders can be motivated to assign high priority to traceability tasks. Finally, further research effort should be devoted to the exploration of traceability challenges in organisational settings. More empirical evidence may boost practitioners to adopt traceability practices.

Our future work consists of a more detailed analysis of the challenges by investigating the interdependencies among the categories. Further, we plan to conduct case studies in order to validate our findings in organisational settings. The findings of this study are part of an ongoing research effort that aims to develop a blockchain-oriented framework for requirements traceability in interorganisational software projects [118].

ACKNOWLEDGEMENT

This work is part of a PhD project, funded by Østfold University College.

ORCID

Mary Sanchez-Gordon  <https://orcid.org/0000-0002-5102-1122>

Ricardo Colomo-Palacios  <https://orcid.org/0000-0002-1555-9726>

REFERENCES

- Pohl, K.: *Requirements Engineering: Fundamentals, Principles, and Techniques*, 1st ed. Springer Publishing Company, Incorporated (2010)
- Regan, G., et al.: The barriers to traceability and their potential solutions: towards a reference framework. In: 2012 38th Euromicro Conference on Software Engineering and Advanced Applications, pp. 319–322. (2012)
- Murugappan, S., Prabha, D.: Requirement traceability for software development lifecycle. *Int. J. Sci. Eng. Res.* 8(5), 1–11 (2017)
- Mustafa, N., Labiche, Y.: The need for traceability in heterogeneous systems: a systematic literature review. In: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), pp. 305–310. (2017)
- Mäder, P., Egyed, A.: Do developers benefit from requirements traceability when evolving and maintaining a software system? *Empir. Softw. Eng.* 20(2), 413–441 (2015)
- Kchaou, D., et al.: Recovering semantic traceability between requirements and design for change impact analysis. *Innov. Syst. Softw. Eng.* 15(2), 101–115 (2019)
- Regan, G., et al.: Medical device standards requirements for traceability during the software development lifecycle and implementation of a traceability assessment model. *Comput. Stand. Interfaces.* 36(1), 3–9 (2013)
- Caffery, F.M., et al.: Medical device software traceability. In: Cleland-Huang, J., Gotel, O., Zisman, A. (eds.) *Software and Systems Traceability*, pp. 321–339. Springer (2012)
- Mäder, P., Olivetto, R., Marcus, A.: Empirical studies in software and systems traceability. *Empir. Softw. Eng.* 22(3), 963–966 (2017)
- 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)
- Gotel, O., et al.: *Traceability fundamentals*. In: Cleland-Huang, J., Gotel, O., Zisman, A. (eds.) *Software and Systems Traceability*, pp. 3–22. Springer (2012)
- Maro, S., et al.: Traceability maintenance: factors and guidelines. In: 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 414–425. (2016)
- Wang, B., et al.: Requirements traceability technologies and technology transfer decision support: a systematic review. *J. Syst. Softw.* 146, 59–79 (2018)
- Sundaram, S.v et al.: Assessing traceability of software engineering artifacts. *Requirements Eng.* 15(3), 313–335 (2010)
- Kong, W.-K., et al.: Process improvement for traceability: a study of human fallibility. In: 2012 20th IEEE International Requirements Engineering Conference (RE), pp. 31–40. (2012)
- Maro, S., et al.: Vetting automatically generated trace links: what information is useful to human analysts? In: 2018 IEEE 26th International Requirements Engineering Conference (RE), pp. 52–63. (2018)
- Winkler, S., von Pilgrim, J.: A survey of traceability in requirements engineering and model-driven development. *Softw. Syst. Model.* 9(4), 529–565 (2010)
- Blauboer, F., Sikkil, K., Aydin, M.N.: Deciding to adopt requirements traceability in practice. In: Krogstie, J., Opdahl, A., Sindre, G. (eds.) *Advanced Information Systems Engineering*, pp. 294–308. Springer (2007)
- Klimpke, L., Hildenbrand, T.: Towards end-to-end traceability: insights and implications from five case studies. In: 2009 Fourth International Conference on Software Engineering Advances, pp. 465–470. (2009)
- Berry, D., et al.: The case for dumb requirements engineering tools. In: Regnell, B., Damian, D. (eds.) *Requirements Engineering: Foundation for Software Quality*, pp. 211–217. Springer (2012)
- Saiedian, H., Kannenberg, A., Morozov, S.: A streamlined, cost-effective database approach to manage requirements traceability. *Softw. Qual. J.* 21(1), 23–38 (2013)
- Niu, N., Wang, W., Gupta, A.: Grey links in the use of requirements traceability. In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 384–395. ACM (2016)
- Gayer, S., et al.: Lightweight traceability for the agile architect. *Computer.* 49(5), 64–71 (2016)
- Fernández, D.M.: Supporting requirements – engineering research that industry needs: The NaPIRE Initiative. *IEEE Softw.* 35(1), 112–116 (2018)
- Regan, G., et al.: Traceability-why do it? In: Mas, A., et al. (eds.) *Software Process Improvement and Capability Determination*, pp. 161–172. Springer (2012)
- Torkar, R., et al.: Requirements traceability: a systematic review and industry case study. *Int. J. Softw. Eng. Knowl. Eng.* 22, 385–433 (2012)
- Maro, S., Steghöfer, J.-P., Staron, M.: Software traceability in the automotive domain: challenges and solutions. *J. Syst. Soft.* 141, 85–110 (2018)
- Gotel, O., et al.: *The Grand Challenge of Traceability (v1. 0)* (2011)
- Ramesh, B.: Factors influencing requirements traceability practice. *Commun. ACM.* 41(12), 37–44 (1998)
- Kannenberg, A., Saiedian, H.: Why software requirements traceability remains a challenge. *J. Defense Softw. Eng.* 22, 14–19 (2009)
- Nair, S., de la Vara, J.L., Sen, S.: A review of traceability research at the requirements engineering conference@21. In: 2013 21st IEEE International Requirements Engineering Conference (RE), pp. 222–229. (2013)
- Wieringa, R., et al.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Eng.* 11(1), 102–107 (2006)
- Kitchenham, B.: *Procedures for Performing Systematic Reviews*, p. 33
- Moreno-Campos, E., et al.: Towards measuring the impact of the ISO/IEC 29110 standard: a systematic review. In: Barafort, B., et al. (eds.) *Systems, Software and Services Process Improvement*, pp. 1–12. Springer (2014)
- Sánchez-Gordón, M., Colomo-Palacios, R.: Taking the emotional pulse of software engineering —A systematic literature review of empirical studies. *Infor. Softw. Technol.* 115, 23–43 (2019)
- Sánchez-Gordón, M., Colomo-Palacios, R.: Characterizing DevOps culture: a systematic literature review. In: Stamelos, I., et al. (eds.) *Software Process Improvement and Capability Determination*, pp. 3–15. Springer International Publishing (2018)
- Kuhrmann, M., Fernández, D.M., Daneva, M.: On the pragmatic design of literature studies in software engineering: an experience-based guideline. *Empir. Softw. Eng.* 22(6), 2852–2891 (2017)
- 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*, pp. 29–38. (2012)
- What have we learned from the challenges of (semi-) automated requirements traceability? A discussion on blockchain applicability. figshare (2020)
- Mäder, P., Gotel, O., Philippow, I.: Enabling automated traceability maintenance through the upkeep of traceability relations. In: Paige, R.F., Hartman, A., Rensink, A. (eds.) *Model Driven Architecture – Foundations and Applications*, pp. 174–189. Springer (2009)
- McCaffery, F., Casey, V.: Med-Trace. In: O'Connor, R.V., et al. (eds.) *Software Process Improvement and Capability Determination*, pp. 208–211. Springer (2011)
- Wohlin, C., Höst, M., Henningsson, K.: Empirical research methods in software engineering. In: Conradi, R., Wang, A.I. (eds.) *Empirical Methods and Studies in Software Engineering: Experiences from ESERNET*, pp. 7–23. Springer (2003)
- Kitchenham, B.: Guidelines for Performing Systematic Literature Reviews in Software Engineering, p. 44

44. Easterbrook, S., et al.: Selecting empirical methods for software engineering research. In: Shull, F., Singer, J., Sjöberg, D.I.K. (eds.) *Guide to Advanced Empirical Software Engineering*, pp. 285–311. Springer (2008)
45. Hong, Y., Kim, M., Lee, S.-W.: Requirements management tool with evolving traceability for heterogeneous artifacts in the entire life cycle. In: 2010 Eighth ACIS International Conference on Software Engineering Research on Management and Applications, pp. 248–255. (2010)
46. Zhou, J., Lu, Y., Lundqvist, K.: A context-based information retrieval technique for recovering use-case-to-source-code trace links in embedded software systems. In: 2013 39th Euromicro Conference on Software Engineering and Advanced Applications, pp. 252–259. (2013)
47. Asuncion, H.U., Taylor, R.N.: Capturing custom link semantics among heterogeneous artifacts and tools. In: Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, pp. 1–5. IEEE Computer Society (2009)
48. Diaz, D. et al.: Using code ownership to improve IR-based traceability link recovery. In: 2013 21st International Conference on Program Comprehension (ICPC), pp. 123–132. (2013)
49. Capobianco, G., et al.: Improving IR-based traceability recovery via noun-based indexing of software artefacts. *J. Softw. Evol. Process.* 25(7), 743–762 (2013)
50. Mahmoud, A., Niu, N.: Supporting requirements traceability through refactoring. In: 2013 21st IEEE International Requirements Engineering Conference (RE), pp. 32–41. (2013)
51. Gervasi, V., Zowghi, D.: Supporting traceability through affinity mining. In: 2014 IEEE 22nd International Requirements Engineering Conference (RE), pp. 143–152. (2014)
52. Borg, M., Gotel, O.C.Z., Wnuk, K.: Enabling traceability reuse for impact analyses: a feasibility study in a safety context. In: 2013 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE), pp. 72–78. (2013)
53. Dietrich, T., Cleland-Huang, J., Shin, Y.: Learning effective query transformations for enhanced requirements trace retrieval. In: 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 586–591. (2013)
54. Pandanaboyana, S., et al.: REquirements TRacing on target (RETRO) enhanced with an automated thesaurus builder: an empirical study. In: 2013 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE), pp. 61–67. (2013)
55. Ziftci, C., Krueger, I.: Tracing requirements to tests with high precision and recall. In: 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pp. 472–475. (2011)
56. Li, Y., Cleland-Huang, J.: Ontology-based trace retrieval. In: 2013 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE), pp. 30–36. (2013)
57. Shin, Y., Cleland-Huang, J.: A comparative evaluation of two user feedback techniques for requirements trace retrieval. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 1069–1074. Association for Computing Machinery (2012)
58. De Lucia, A., et al.: Applying a smoothing filter to improve IR-based traceability recovery processes: an empirical investigation. *Infor. Softw. Technol.* 55(4), 741–754 (2013)
59. Gibiec, M., Czauderna, A., Cleland-Huang, J.: Towards mining replacement queries for hard-to-retrieve traces. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, pp. 245–254. Association for Computing Machinery (2010)
60. Chen, X., Grundy, J.: Improving automated documentation to code traceability by combining retrieval techniques. In: 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pp. 223–232. (2011)
61. Tsuchiya, R., et al.: Interactive recovery of requirements traceability links using user feedback and configuration management logs. In: Zdravkovic, J., Kirikova, M., Johannesson, P. (eds.) *Advanced Information Systems Engineering*, pp. 247–262. Springer International Publishing (2015)
62. Wang, W., et al.: Enhancing automated requirements traceability by resolving polysemy. In: 2018 IEEE 26th International Requirements Engineering Conference (RE), pp. 40–51. (2018)
63. Ghannem, A., et al.: Search-based requirements traceability recovery: a multi-objective approach. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1183–1190. (2017)
64. Lucassen, G., et al.: Behavior-driven requirements traceability via automated acceptance tests (REW). In: 2017 IEEE 25th International Requirements Engineering Conference Workshops, pp. 431–434. (2017)
65. Haidrar, S., Anwar, A., Roudies, O.: Towards a generic framework for requirements traceability management for SysML language. In: 2016 4th IEEE International Colloquium on Information Science and Technology (CiSt), pp. 210–215. (2016)
66. Chen, L., et al.: Enhancing unsupervised requirements traceability with sequential semantics. In: 2019 26th Asia-Pacific Software Engineering Conference (APSEC), pp. 23–30. (2019)
67. Mordinyi, R., Biffl, S.: Exploring traceability links via issues for detailed requirements coverage reports. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), pp. 359–366. (2017)
68. Wang, W., et al.: Automatically tracing dependability requirements via term-based relevance feedback. *IEEE Trans. Ind. Infor.* 14(1), 342–349 (2018)
69. Pakdeetrakulwong, U., Wongthongtham, P., Khan, N.: An ontology-based multi-agent system to support requirements traceability in multi-site software development environment. In: Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference, pp. 96–100. ACM (2015)
70. Guo, J., Cheng, J., Cleland-Huang, J.: Semantically enhanced software traceability using deep learning techniques. In: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE) (ICSE), pp. 3–14. (2017)
71. Panichella, A., De Lucia, A., Zaidman, A.: Adaptive user feedback for IR-based traceability recovery. In: 2015 IEEE/ACM 8th International Symposium on Software and Systems Traceability, pp. 15–21. (2015)
72. Ali, N., et al.: Exploiting Parts-of-Speech for effective automated requirements traceability. *Infor. Softw. Technol.* 106, 126–141 (2019)
73. Mäder, P., Gotel, O.: Towards automated traceability maintenance. *J. Syst Softw.* 85(10), 2205–2227 (2012)
74. Rempel, P., Mäder, P., Kuschke, T.: An empirical study on project-specific traceability strategies. In: 2013 21st IEEE International Requirements Engineering Conference (RE), pp. 195–204. (2013)
75. Mader, P., Gotel, O., Philippow, I.: Motivation matters in the traceability trenches. In: 2009 17th IEEE International Requirements Engineering Conference, pp. 143–148. (2009)
76. Wohlrab, R., et al.: Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture. *Requirements Eng.* (2018)
77. Sanchez, B.A.: Context-aware traceability across heterogeneous modelling environments. In: Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, pp. 174–179. ACM (2018)
78. Amalfitano, D., et al.: Using tool integration for improving traceability management testing processes: an automotive industrial experience. *J. Softw. Evol. Process.* 31(6), e2171 (2019)
79. Panis, M.C.: Successful deployment of requirements traceability in a commercial engineering organization really. In: 2010 18th IEEE International Requirements Engineering Conference, pp. 303–307. (2010)
80. Rahimi, M., Cleland-Huang, J.: Evolving software trace links between requirements and source code. *Empir. Softw. Eng.* 23(4), 2198–2231 (2018)
81. Mills, C., Escobar-Avila, J., Haiduc, S.: Automatic traceability maintenance via machine learning classification. In: 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 369–380. (2018)
82. Figueiredo, M.C., de Souza, C.R.B.: Wolf supporting impact analysis activities in distributed software development. In: 2012 5th

- International Workshop on Co-operative and Human Aspects of Software Engineering (CHASE), pp. 40–46. (2012)
83. Helming, J., et al.: Traceability-based change awareness. In: Schürr, A., Selic, B. (eds.): *Model Driven Engineering Languages and Systems*, pp. 372–376. Springer (2009)
 84. Mäder, P., et al.: Strategic traceability for safety-critical projects. *IEEE Softw.* 30(3), 58–66 (2013)
 85. 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)
 86. Cuddeback, D., Dekhtyar, A., Hayes, J.: Automated requirements traceability: the study of human analysts. In: 2010 18th IEEE International Requirements Engineering Conference, pp. 231–240. (2010)
 87. Dekhtyar, A., et al.: On human analyst performance in assisted requirements tracing: statistical analysis. In: 2011 IEEE 19th International Requirements Engineering Conference, pp. 111–120. (2011)
 88. Kong, W.-K., et al.: How do we trace requirements: an initial study of analyst behavior in trace validation tasks. In: *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 32–39. ACM (2011)
 89. Niu, N., et al.: Departures from optimality: understanding human analyst's information foraging in assisted requirements tracing. In: 2013 35th International Conference on Software Engineering (ICSE), pp. 572–581. (2013)
 90. Cuddeback, D., et al.: Towards overcoming human analyst fallibility in the requirements tracing process: NIER track. In: 2011 33rd International Conference on Software Engineering (ICSE), pp. 860–863. (2011)
 91. Borg, M., Pfahl, D.: Do better IR tools improve the accuracy of engineers' traceability recovery? In: *Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering*, pp. 27–34. Association for Computing Machinery (2011)
 92. Hayes, J.H., et al.: Effective use of analysts' effort in automated tracing. *Requirements Eng.* 23(1), 119–143 (2018)
 93. Wang, W., et al.: Tagging in assisted tracing. In: 2015 IEEE/ACM 8th International Symposium on Software and Systems Traceability, pp. 8–14. (2015)
 94. Casey, V., Caffery, F.M.: A lightweight traceability assessment method for medical device software. *J. Softw. Evol. Process.* 25(4), 363–372 (2013)
 95. Rempel, P., Mäder, P.: A quality model for the systematic assessment of requirements traceability. In: 2015 IEEE 23rd International Requirements Engineering Conference (RE), pp. 176–185. (2015)
 96. Regan, G., et al.: Assessing traceability-practical experiences and lessons learnt. *J. Softw. Evol. Process.* 27(8), 591–601 (2015)
 97. Ståhl, D., Hallén, K., Bosch, J.: Continuous integration and delivery traceability in industry: needs and practices. In: 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 68–72. (2016)
 98. Espinoza, A., Garbajosa, J.: A study to support agile methods more effectively through traceability. *Innov. Syst. Softw. Eng.* 7(1), 53–69 (2011)
 99. Rubasinghe, I., Meedeniya, D., Perera, I.: Automated inter-artefact traceability establishment for DevOps practice. In: 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), pp. 211–216. (2018)
 100. Murtazina, M.S., Avdeenko, T.V.: An ontology-based approach to support for requirements traceability in agile development. *Procedia Comput. Sci.* 150, 628–635 (2019)
 101. Elamin, R., Osman, R.: Towards requirements reuse by implementing traceability in agile development. In: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), pp. 431–436. (2017)
 102. Furtado, F., Zisman, A.: Trace++: a traceability approach to support transitioning to agile software engineering. In: 2016 IEEE 24th International Requirements Engineering Conference (RE), pp. 66–75 (2016)
 103. Rempel, P., et al.: Requirements traceability across organizational boundaries - a survey and taxonomy. In: Doerr, J., Opdahl, A.L. (eds.) *Requirements Engineering: Foundation for Software Quality*, pp. 125–140. Springer (2013)
 104. Fernández, D.M., Wagner, S.: Naming the pain in requirements engineering: design of a global family of surveys and first results from Germany. In: *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering - EASE '13*, p. 183 (2013)
 105. Fernando Capretz, L.: Bringing the human factor to software engineering. *IEEE Softw.* 31(2), 104 (2014)
 106. Colomo-Palacios, R.: Cross fertilization in software engineering. In: Yilmaz, M., et al. (eds.) *Systems, Software and Services Process Improvement*, pp. 3–13. Springer International Publishing (2020)
 107. 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)
 108. Swan, M.: *Blockchain: Blueprint for a New Economy*. O'Reilly Media, Inc. (2015)
 109. Tschorsch, F., Scheuermann, B.: Bitcoin and beyond: a technical survey on decentralized digital currencies. *IEEE Commun. Surv. Tutorials.* 18(3), 2084–2123 (2016)
 110. Zheng, Z., et al.: Blockchain challenges and opportunities: a survey. *Int. J. Web Grid Serv.* 14(4), 352–375 (2018)
 111. Demi, S., Colomo-Palacios, R., Sánchez-Gordón, M.: Software engineering applications enabled by blockchain technology: a systematic mapping study. *Appl. Sci.* 11(7), 2960 (2021)
 112. Vacca, A., et al.: A systematic literature review of blockchain and smart contract development: techniques, tools and open challenges. *J. Syst. Softw.* 174, 110891 (2021)
 113. Yilmaz, M., et al.: Applying blockchain to improve the integrity of the software development process. In: Walker, A., O'Connor, R.V., Messnarz, R. (eds.) *Systems, Software and Services Process Improvement*, pp. 260–271. Springer International Publishing (2019)
 114. Singi, K., et al.: 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)
 115. Bose, R.P.J.C., et al.: BLINKER: a blockchain-enabled framework for software provenance. In: 2019 26th Asia-Pacific Software Engineering Conference (APSEC), pp. 1–8. (2019)
 116. Bose, R.P.J.C., et al.: Framework for trustworthy software development. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW), pp. 45–48. (2019)
 117. Damian, D., Chisan, J.: An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management. *IEEE Trans. Softw. Eng.* 32(7), 433–453 (2006)
 118. Demi, S.: Blockchain-oriented requirements engineering: a framework. In: 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 428–433. (2020)

How to cite this article: Demi, S., Sanchez-Gordon, M., Colomo-Palacios, R.: What have we learnt from the challenges of (semi-) automated requirements traceability? A discussion on blockchain applicability. *IET Soft.* 15(6), 391–411 (2021). <https://doi.org/10.1049/sfw.2.12035>

APPENDIX: LIST OF PRIMARY STUDIES

ID	Study reference	Title
PS1	[86]	Automated requirements traceability: The study of human analysts
PS2	[87]	On human analyst performance in assisted requirements tracing: Statistical analysis
PS3	[15]	Process improvement for traceability: A study of human fallibility
PS4	[88]	How do we trace requirements: an initial study of analyst behavior in trace validation tasks
PS5	[89]	Departures from optimality: Understanding human analyst's information foraging in assisted requirement tracing
PS6	[90]	Towards overcoming human analyst fallibility in requirements tracing process: NIER track
PS7	[16]	Vetting automatically generated trace links: What information is useful to human analysts
PS8	[92]	Effective use of analysts' effort in automated tracing
PS9	[76]	Collaborative traceability management: A multiple case study from the perspectives of organization, process and culture
PS10	[103]	Requirements traceability across organizational boundaries: A survey and taxonomy
PS11	[7]	Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model
PS12	[94]	A lightweight traceability assessment method for medical device software
PS13	[73]	Towards automated traceability maintenance
PS14	[12]	Traceability maintenance: factors and guidelines
PS15	[14]	Assessing traceability of software engineering artefacts
PS16	[45]	Requirements management tool with evolving traceability for heterogeneous artefacts in the entire lifecycle
PS17	[47]	Capturing custom link semantics among heterogeneous artefacts and tools
PS18	[61]	Interactive recovery of requirements traceability links using user feedback and configuration management logs
PS19	[77]	Context-aware traceability across heterogeneous modelling
PS20	[97]	Continuous integration and delivery traceability in industry: needs and practices
PS21	[26]	Requirements traceability: A systematic review and industry case study
PS22	[48]	Using code ownership to improve IR-based traceability link recovery
PS23	[22]	Grey links in the use of requirements traceability
PS24	[62]	Enhancing automated requirements traceability by resolving polysemy
PS25	[93]	Tagging in assisted tracing
PS26	[49]	Improving IR-based traceability recovery via noun-based indexing of software artefacts
PS27	[84]	Strategic traceability for safety-critical projects
PS28	[50]	Supporting requirements traceability through refactoring
PS29	[51]	Supporting traceability through affinity mining
PS30	[52]	Enabling traceability reuse for impact analysis: A feasibility study in a safety context
PS31	[63]	Search-based requirements traceability recovery: A multi-objective approach
PS32	[99]	Automated inter-artefact traceability establishment for DevOps practice
PS33	[27]	Software traceability in the automotive domain: Challenges and solutions
PS34	[64]	Behavior-driven requirements traceability via automated acceptance tests
PS35	[85]	Interactive traceability links visualization using hierarchical trace map

APPENDIX (Continued)

ID	Study reference	Title
PS36	[53]	Learning effective query transformations for enhanced requirements trace retrieval
PS37	[65]	Towards a generic framework for requirements traceability management for SysML language
PS38	[66]	Enhancing unsupervised requirements traceability with sequential semantics
PS39	[54]	Requirements Tracing on Target (RETRO) enhanced with an automated thesaurus builder
PS40	[67]	Exploring traceability links via issues for detailed requirements coverage reports
PS41	[55]	Tracing requirements to tests with high precision and recall
PS42	[46]	A context-based information retrieval technique for recovering use-case-to-source-code trace links in embedded software systems
PS43	[68]	Automatically tracing dependability requirements via term-based relevance feedback
PS44	[98]	A study to support agile methods through traceability
PS45	[56]	Ontology-based trace retrieval
PS46	[69]	An ontology-based multi-agent system to support requirements traceability in multi-site software development environment
PS47	[100]	An ontology-based approach to support for requirements traceability in agile development
PS48	[101]	Towards requirements reuse by implementing traceability in agile development
PS49	[74]	An empirical study on project-specific traceability strategies
PS50	[80]	Evolving software trace links between requirements and source code
PS51	[79]	Successful deployment of requirements traceability in a commercial engineering organization... really
PS52	[75]	Motivation matters in the traceability trenches
PS53	[19]	Towards end-to-end traceability: Insights and implications from five case studies
PS54	[83]	Traceability-based change awareness
PS55	[23]	Lightweight traceability for the agile architect
PS56	[82]	Wolf: Supporting impact analysis activities in distributed software development
PS57	[81]	Automatic traceability maintenance via machine learning classification
PS58	[95]	A quality model for the systematic assessment of requirements traceability
PS59	[21]	A streamlined, cost-effective database approach to manage requirements traceability
PS60	[57]	A comparative evaluation of two user-feedback techniques for requirements trace retrieval
PS61	[58]	Applying a smoothing filter to improve IR-based traceability recovery processes: An empirical investigation
PS62	[59]	Towards mining replacement queries for hard-to-retrieve traces
PS63	[70]	Semantically enhanced software traceability using deep learning techniques
PS64	[60]	Improving automated documentation to code traceability by combining retrieval techniques
PS65	[91]	Do better IR tools improve the accuracy of engineers' traceability recovery?
PS66	[71]	Adaptive user feedback for IR-based traceability recovery
PS67	[72]	Exploiting parts-of-speech for effective automated requirements traceability
PS68	[102]	Trace++: A traceability approach to support transitioning to agile software engineering
PS69	[96]	Assessing traceability- practical experiences and lessons learnt
PS70	[78]	Using tool integration for improving traceability management testing processes: An automotive industrial experience




PAPER III

S. Demi, R. Colomo-Palacios, and M. Sanchez-Gordon, "*Software Engineering Applications Enabled by Blockchain Technology: A Systematic Mapping Study*," *Applied sciences*, vol. 11, no. 7, p. 2960, 2021.

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

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

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].

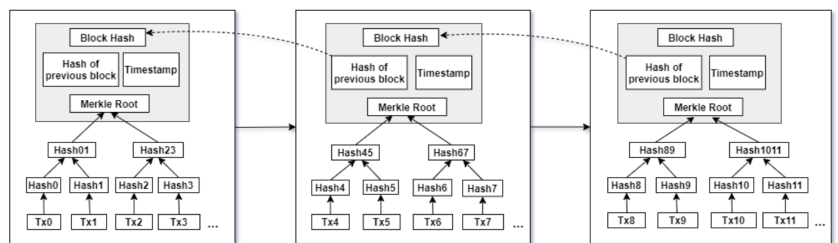


Figure 1. Blockchain structure, adapted from [20].

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 contracts 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.

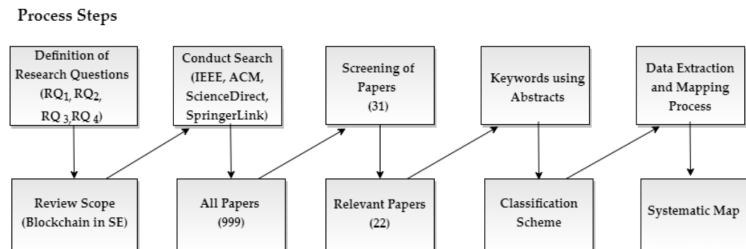


Figure 2. Overview of the systematic mapping study, adapted from [28].

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. RQ₁ *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. RQ₂ *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. RQ₃ *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. RQ₄ *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 (RQ₁)

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.

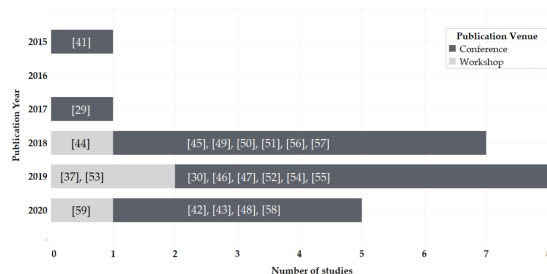


Figure 3. Distribution of studies based on publication year and publication venue.

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.

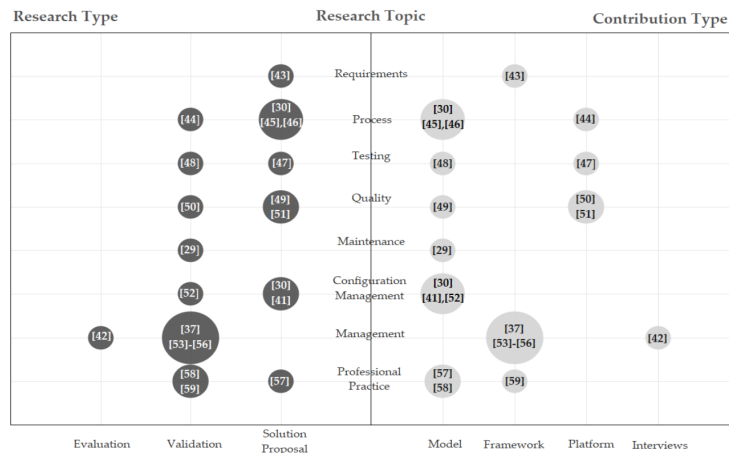


Figure 4. Visualization of our systematic map in the form of a bubble plot.

4.2. Blockchain Uses in Software Engineering Reported in Literature (RQ₂)

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 (RQ₃)

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.

	Bitcoin	[41]
Platforms	Ethereum	[29], [37], [44] *, [52], [53] ^Δ , [54] [¥] , [55] [†] , [59]
	Hyperledger Fabric	[48], [49], [56] [◇] , [58]
	Others	Susereum [50]

* ChainSoft, ^Δ CAG, [¥] Blinker, [†] ShIFt, [◇] 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 (RQ₄)

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 initiators 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 practices 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 pre-defined 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	ShIPt—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. Niranjnamurthy, 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, ON, Canada, 29–31 October 2018; pp. 315–322.
51. Lin, Y.; Qi, Z.; Wu, H.; Yang, Z.; Zhang, J.; Wenyan, 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 (QVMMMA) 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](#)]






PAPER IV

S. Demi, M. Sánchez-Gordón, and R. Colomo-Palacios, "*A Blockchain-Enabled Framework for Requirements Traceability*," in *Systems, Software and Services Process Improvement: 28th European Conference, EuroSPI 2021, Krems, Austria, September 1–3, 2021, Proceedings*, pp. 3–13, Springer, 2021.



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

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:

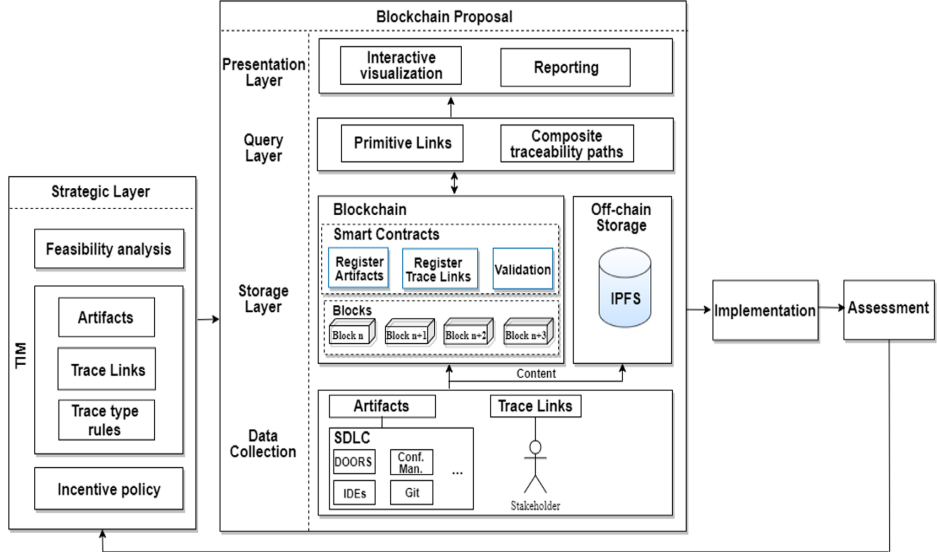


Fig. 1. Blockchain-enabled requirements traceability framework

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.

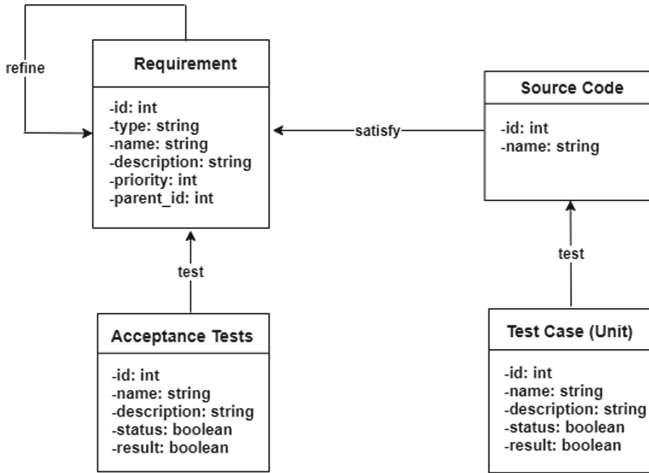


Fig. 2. A simple traceability information model

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 accessibility, 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 use, 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>



PAPER V

S. Demi, M. Sánchez-Gordón, and M. Kristiansen, "*Blockchain for Requirements Traceability: A Qualitative Approach*," *Journal of Software: Evolution and Process*, p. e2493, 2022.

Blockchain for requirements traceability: A qualitative approach

Selina Demi  | Mary Sánchez-Gordón  | Monica Kristiansen

Østfold University College, Norway

Correspondence

Mary Sánchez-Gordón, Østfold University College, Norway.

Email: mary.sanchez-gordon@hiof.no

Funding information

Østfold University College

Abstract

Blockchain technology has emerged as a “disruptive innovation” that has received significant attention in academic and organizational settings. However, most of the existing research is focused on technical issues of blockchain systems, overlooking the organizational perspective. This study adopted a grounded theory to unveil the blockchain implementation process in organizations from the lens of blockchain experts. The results revealed three main categories: key activities, success factors, and challenges related to blockchain implementation in organizations, the latter being identified as the core category, along with 17 other concepts. Findings suggested that the majority of blockchain projects stop at the pilot stage and outlined organizational resistance to change as the core challenge. According to the experts, the following factors contribute to the organizational resistance to change: innovation–production gap, conservative management, and centralized mentality. The study aims to contribute to the existing blockchain literature by providing a holistic and domain-agnostic view of the blockchain implementation process in organizational settings. This can potentially encourage the development and implementation of blockchain solutions and guide practitioners who are interested in leveraging the inherent benefits of this technology. In addition, the results are used to improve a blockchain-enabled requirements traceability framework proposed in our previous paper.

KEYWORDS

blockchain technology, grounded theory, interorganizational software projects, requirements traceability

1 | INTRODUCTION

Blockchain (BC) has been considered a cutting-edge technology with the potential to disrupt conventional domains and business models,¹ as pervasively as the Internet had done.² The Internet changed humans' understanding of time and space by intensifying social relations, creating a global ecosystem.³ Within this global ecosystem, BC is transforming the nature of human relations and organizations by enabling smart contracts (SC) that ensure trust among individuals and organizations.³ From an architectural perspective, BC is a distributed ledger technology that stores all committed transactions in an ever-growing chain of blocks.⁴ The fundamental feature of BC is peer-to-peer data sharing and storage, removing the need to entrust central authorities for the maintenance of the ledger.⁵ Although BC research is on the rise,⁶ most of this research is focused on a technical perspective and takes a simplistic view on organizational issues.⁷ Our preliminary research suggested little empirical evidence on

This is an open access article under the terms of the [Creative Commons Attribution-NonCommercial-NoDerivs](https://creativecommons.org/licenses/by-nc-nd/4.0/) License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2022 The Authors. Journal of Software: Evolution and Process published by John Wiley & Sons Ltd.

how BC is implemented in organizational settings. Beck and Müller-Bloch⁸ concluded that BC is a radical innovation, and as such, organizations engage with this technology by means of three main processes: discovery, incubation, and acceleration. Similarly, Dozier and Montgomery⁹ used a grounded theory (GT) approach to unveil the BC evaluation process. To explain the evaluation process, the authors presented the proof-of-value model that is composed of three main activities: understand, organize, and test. Despite these valuable efforts, further empirical evidence is needed to enhance the understanding of the implementation process of BC in organizations.

Inherent benefits of BC, such as disintermediation, automation, trust, cost reduction, and non-repudiation have generated growing interest¹⁰ across a variety of industry sectors ranging from financial services to manufacturing and public services.⁷ Recently, an increasing number of BC applications has been noticed in the software engineering (SE) landscape.¹¹ BC has also been proposed to keep track of artifacts and trace links created by distributed stakeholders in interorganizational software projects.¹² This study is an extension of our previous study presented in EuroSPI conference.¹² The authors aim to improve the proposed BC-enabled requirements traceability (RT) framework by using categories and concepts grounded in data. GT was identified as the most suitable approach to unveil these categories and concepts, given the novel and multi-faceted nature of BC. Data were collected through semi-structured and in-depth interviews with BC experts and analyzed by means of GT coding techniques. The results revealed the core category, *blockchain implementation in organizations*; three related categories, *key activities*, *success factors* and *challenges*; and 17 concepts to further expound on the categories.

The contributions of this study lie in two main dimensions: (i) providing evidence on the implementation process of BC in organizational settings from the lens of BC experts. This evidence may serve as a guide for researchers and practitioners who are interested in this technology, enhance the comprehension of this technology, and consequently encourage the development and implementation of BC solutions. Another is (ii) improving our previous BC-enabled RT framework¹² by supporting it with categories that are grounded in empirical data.

The sections of this study are as follows: Section 2 presents the extant literature in the fields of BC-enabled SE, and RT. Section 3 describes the research approach followed in this study, the experts' selection process, data collection, and analysis. Section 4 provides a thorough description and explanation of the key results of the study. Section 5 discusses the validity of the results and limitations of the study. In addition, this section puts the results in the context of a specific use case on the application of BC for RT in interorganizational software projects. Finally, Section 6 concludes the study and presents a set of potential future research directions.

2 | BACKGROUND

2.1 | BC for SE

BC has made the concept of "shared registry" possible for a variety of application domains ranging from cryptocurrencies to potentially any system that requires decision-making to be decentralized, reliable, and automated in a multi-stakeholder environment.⁵ Recently, BC has captured the attention of SE researchers who advocate for the cross-fertilization of SE with BC technologies.^{13,14} Our recent systematic mapping study on this topic revealed an increasing trend of studies that have used BC in the SE landscape since 2018.¹¹ A set of these studies is presented as follows. Lenarduzzi et al.¹⁵ proposed the automation of the acceptance phase and the payment to developers by means of SC. SC are created by the customer, and then the product owner registers the following artifacts: user stories, acceptance tests that are executed by developers, and the hash of expected output. The hash of the output is assessed against the hash of the expected output and once all the tests pass, Ethers are allocated to the address of the developer. According to the authors of this study, BC can potentially transform other phases of the software lifecycle that currently depend on human rationale.

Beller and Hejderup¹⁶ introduced two BC-enabled models: (i) a decentralized continuous integration (CI) model that aims to replace the conventional Travis CI. In this model, developers enter builds and their respective rewards to the distributed network and interested entities perform the builds. Additionally, (ii) a user-run package management system as opposed to centralized package management systems allows entities to propose new packages and validate others' work.

Yilmaz et al.¹⁷ focused on improving the integrity of large-scale agile software development. The authors proposed a BC model that considers developers who develop code as miners and testers as the validators of the work performed by developers.

Bose et al.¹⁸ proposed Blinker, a BC-enabled framework for trusted software provenance. The framework monitors and captures provenance data that are generated from various tools used throughout the software lifecycle. The data are modeled according to standard provenance model specification. SC are also created to validate provenance data by voting mechanisms and for compliance checking.

Yau and Patel¹⁹ used BC to ensure trusted coordination in complex and collaborative software development. In their proposal, software teams produce software specifications for different software lifecycle phases, such as requirements, implementation and testing in {key, value} format. These specifications are then evaluated against the output generated by the other teams.

Singi et al.²⁰ presented a token-based incentive framework that uses BC and SC to provide transparent and reliable incentives to those software engineers who contribute to any activities of the software lifecycle. The framework entails capturing incentive policies and their associated

data, monitoring events, and recording events data on the distributed ledger. These data are then assessed against incentive policies and accordingly, incentives are distributed in the form of persistent wallet tokens in an automatic fashion by means of SC.

These studies focused on improving different aspects of the software lifecycle and contributed to software being built in a reliable, transparent, and auditable manner.²⁰ Despite these promising contributions, the BC-enabled SE field is still emerging and further research efforts are required.¹¹ In this study, the authors aim to contribute to this field by presenting a BC-enabled framework for RT in interorganizational software projects.

2.2 | Requirements traceability

As software-intensive systems are becoming increasingly important in industrial projects, and more innovative and high-quality systems are required to be brought to market in a quick fashion, there is a need for efficient requirements engineering.²¹ Within the requirements engineering community, RT has gained importance, as a quality attribute for software for three decades now.^{22,23} The foundational work in this field has been carried out by Gotel and Finkelstein²⁴ with their seminal study *An analysis of the requirements traceability problem*. The authors of this study defined RT as “the ability to describe and follow the life of a requirement, in both a forwards and backwards direction” At a fundamental scope, traceability refers to the ability to relate artifacts’ data and examine these relations.²⁵ These relations provide valuable information that contribute to a variety of software and systems engineering activities, such as software maintenance,²⁶ change and defect management,^{27,28} and project management.²⁸ Ultimately, RT ensures visibility into required elements of the development process, which leads to a better understanding of the system under development.²⁵

The theoretical importance of RT would suggest a widespread use of RT in practice. However, in practice, RT has been perceived as an optional task of low priority and performed by means of ad-hoc individual efforts.^{24,29} Previous studies have proposed the inclusion of gamification elements²⁹ and voting features into traceability tools.³⁰ The former aims to enhance the motivation of developers to engage in traceability tasks, and the latter enables stakeholders to identify incorrect trace links or agree on related artifacts as a result of a joint effort.

The increased need for complex and large-scale software has paved the way for the distributed development paradigm, that is, development performed by cross-organizational and distributed teams.¹² According to Maro et al.,²⁹ this development paradigm complicates RT as distributed teams need to share software artifacts. These artifacts can be stored in a centralized data storage and accessed by distributed and diverse teams.²⁹ However, these teams cannot always trust artifacts provided by the other teams as competitors and malicious entities may be involved in the collaboration.³¹ The involvement of third-party vendors raises even more significant trust concerns among the participating entities.¹⁹ The distance produced by participating entities in interorganizational software projects has been acknowledged also by Rempel et al.³² According to the authors, traceability may contribute to bridging this distance. However, achieving complete RT in such environments is far from being trivial, due to three main problem areas: (i) different organizational background of participating entities lead to the use of a diverse set of tools and methodologies that reside within organizational boundaries³²; (ii) contradicting organizational objectives of entities involved in the project, for instance, each entity may create trace links that are not compatible with the other entities in terms of trace links types or granularity;³⁰ and (iii) organizational boundaries may lead to restricted access to artifacts created by the other participating entities, due to confidentiality constraints.^{30,32} Entities may have the right to access only a small subset of the entire set of artifacts generated throughout the software development lifecycle (SDLC). This small subset of artifacts is not sufficient to enable complete RT.³² To address these challenges, this study proposes the use of BC technology as a viable technical solution with the potential to serve as a shared, trusted, and auditable traceability knowledge base.

3 | RESEARCH METHODOLOGY

3.1 | Research approach

This study adopted a qualitative approach by conducting semi-structured interviews with BC experts. BC experts were interviewed in order to provide a comprehensive, holistic, and domain-agnostic overview of the implementation of BC technology in organizational settings. The empirical work in this phase was conducted by applying traditional GT. Although a variety of methodological genres exist, researchers seem to unanimously agree that GT is a process or method by which conceptual frameworks or theories are generated from inductive analysis of data.^{33–35}

In this empirical study, GT was selected as the most suitable research methodology for three main reasons. First, factors influencing the implementation of BC in organizations go beyond technical aspects. The underlying socio-technical nature of this novel technology makes it suitable for the application of GT. Second, GT contributes to investigate a novel multifaced phenomenon in detail.³⁶ Despite the fact that the application of BC in various domains has been explored recently, BC research remains at an early stage in terms of theoretical and empirical grounded work and methodological diversity.³ Lastly, GT is appropriate when researchers do not have an upfront hypothesis; instead of that, they aim to construct a theory grounded in the data. In this study, GT was adopted to derive constructs grounded in the data which will be further used to

improve the initial BC-enabled RT framework.¹² In addition, the study followed the GT guidelines by focusing on a wide area such as BC rather than a specific research problem.³⁷

Although GT has been used vastly in social studies, applications of this method have been noticed also in human-related aspects of SE.^{38,39} Figure 1 depicts the GT stages that are used in this study, adapted from Hoda et al.³⁸

3.2 | Data collection

Data collection in GT is an ongoing activity to achieve theoretical saturation that refers to the point at which no new information is being acquired.³³ The initial data collection process is aimed to identify the main concepts that are then used to decide on the data to be collected in the next stages of the process.⁴⁰ This approach is referred to as theoretical sampling by Glaser⁴⁰ and is adopted by our study. The following sections describe the experts selection and interviewing processes.

3.2.1 | Experts selection

Experts should be selected so that the most credible and accurate judgments are provided. To govern the selection process, a set of criteria was formulated by considering the recommendations of NUREG-1150 cited in Li and Smidts⁴¹: demonstrated experience related to the topic of interest by publications, consulting firms, laboratories, or government agencies; diverse background and affiliations; and willingness to be elicited in accordance to the designed methodology. Bearing these guidelines in mind, Fehring's⁴² experts selection criteria for BC was adapted. Although Fehring's⁴² criteria were initially designed to select nurses to validate nursing diagnoses, they have also been tailored to the software domain.⁴³ Table 1 depicts the criteria for selecting BC experts, and Table 2 presents the characterization of experts.

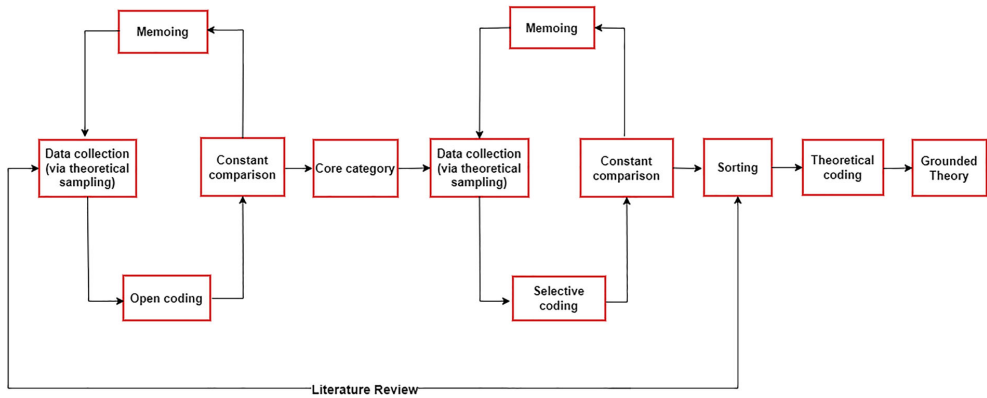


FIGURE 1 Grounded theory stages, adapted from Hoda et al.³⁸

TABLE 1 Selection criteria for blockchain experts, adapted from Fehring⁴²

Experts' selection criteria	Scores
C1.Over 2 years of experience in BC	04
C2.Academic experience (teaching, supervision of bachelor, master, PhD students) in BC	04
C3.PhD in BC	03
C4.Published journal articles in the field of BC	03
C5.Published conference/workshop/symposium articles/book chapters/reports in BC	02
C6.Master's degree and/or thesis in BC	01

Abbreviation: BC, blockchain.

TABLE 2 Profiling experts

Expert	+2 years of experience	Academic experience	PhD	Journal articles	Other publications	Master thesis	Score	Job position	Project domain	Country
EX1	X				X		6	Business development director	e-health, energy, supply chain	Norway
EX2	X				X		6	Solutions architect	supply chain	US
EX3	X				X		6	BC software engineer ^a	all domains	Portugal
EX4	X				X	X	7	Chief executive officer	governments, banking, e-health, energy, insurance	Norway
EX5	X				X	X	7	Researcher	marketing industry, education	Sweden
EX6		X			X		6	Associate professor	public sector	Sweden
EX7	X			X	X		9	Chief technology officer	all domains	Spain
EX8	X				X		6	Head of BC and Data strategy	banking	Norway
EX9		X		X	X		12	Researcher	e-health	Norway
EX10	X			X			7	Chief executive officer	all domains	Spain
N (%)	8 (80%)	2 (20%)	1 (10%)	3 (30%)	9 (90%)	2 (20%)	Avg. (7.2)			

Abbreviation: BC, blockchain.

^aBC technical and business consultant.

The participants of this study were considered BC experts if they were evaluated with a total score of 5 or higher. Forty-five experts in the domains related to this study were identified as potential candidates by means of the personal network of the authors of this study. This sampling approach has been referred to as purposive sampling and has been mainly used in qualitative and interpretive research.⁴⁴ Purposive sampling allows the authors to exercise expert judgment.⁴⁵ To apply the selection criteria presented in Table 1, candidates' LinkedIn profiles, personal websites, and company/university webpage were thoroughly analyzed. Consequently, a set of 35 experts with a score of 5 or higher were selected and contacted by email. The final set of 10 experts was composed based on their availability and interest to participate in this study. A detailed description of the project was sent to the final set of experts prior to their participation in the study. In addition, an informed consent that outlined the implications of experts' participation in this study, their voluntariness, and anonymity rights was sent out to each of the experts (The informed consent is available online⁴⁶). To ensure anonymity, experts of this study are referred to by IDs (EXi for i^{1,10}). Table 2 presents the individual and mean scores of the final set of experts, along with other characteristics. The data indicate that the majority of the selected experts (80%) have over 2 years of practical experience with BC, although only two experts present academic experience and 90% of the experts have published conference/workshop papers, book chapters, or reports on BC-related topics.

3.2.2 | Interviews

The authors collected data by carrying out semi-structured interviews with BC experts. Semi-structured interviews were chosen due to their ability to provide rich contextual information, and to allow for flexibility and improvisation.⁴⁷ This technique enables two-way communication between the interviewer and interviewee, which makes the communication more personal and allows to uncover relevant information for the study.⁴⁸ The interviews were conducted via Zoom and lasted from 40 to 95 min. It is noteworthy that interviews were recorded with the interviewees' formal consent. Relying on GT principles,⁴⁰ the authors asked general questions about the implementation of BC in organizational settings in the first interviews. The transcribed data of these interviews were analyzed in an iterative fashion and used to formulate questions about specific items in the following interviews.

The interviews started with questions regarding the interviewee's experience related to BC and a brief description of the projects they participated. Next, interviewees were asked a set of questions regarding the implementation process of BC in organizational settings, implementation challenges and success factors, factors to be considered when planning to implement BC, and the selection process of the best fitting BC platform. A sample of the questions can be found in an online repository.⁴⁶ The data collected from these interviews were analyzed by using GT coding techniques as explained in Section 3.3, and the BC-enabled RT framework was updated accordingly (see Section 5.2). It is noteworthy that the BC experts were not provided with the framework beforehand to avoid potential biases.

3.3 | Data analysis

The data analysis approach adopted in this study relies on the traditional GT approach associated with Glaser.³⁴ Qualitative research is characterized by an abundance of data, which are difficult to be managed manually. To minimize human error when collecting and analyzing large amounts of data, qualitative researchers use Computer-Assisted Qualitative Data Analysis Software (CAQDAS). In this study, the authors used NVivo software to handle a large amount of qualitative data throughout the whole GT process. This software has been used by other GT studies in SE, for example, Javdani Gandomani and Ziaei Nafchi.⁴⁹ In what follows, the GT data analysis process is described.

- i. *Open coding*. In this phase, transcripts were analyzed in order to comprehend the context under study.⁴⁰ The analysis aimed to identify key points. Once a key point was identified, a code was assigned to that specific key point.⁴⁰ Furthermore, the code was compared with previously identified codes in the same transcript and previous transcripts. This process has been referred to as constant comparison and enables high-level abstraction and the identification of concepts and categories.⁴⁰ The emergent categories will be used as the basis to generate the final theory.
- ii. *Core category*. The open coding process terminates with the identification of the core category. The core category reflects the main concerns of the interviewees.⁴⁰ According to Glaser,⁴⁰ the core category should fulfill a set of criteria: being central, being related to the other categories meaningfully, and accounting for the majority of data variations. The core category can be identified by using the constant comparison technique on the categories and by identifying their relationships. After the first four interviews, the authors perceived "feasibility analysis" as the main category, as it was related to the other categories meaningfully. However, a more detailed analysis indicated that this category did not fit the following criteria: being central and account for the majority of data variations. Therefore, the authors continued the iterative constant comparison process and consequently, concluded that the core category was "BC implementation in organizations." The identification of the core category terminates the open coding process and paves the way to selective coding. Selective coding in GT entails coding only the core category, along with its related categories.⁴⁰

- iii. *Theoretical memoing.* The authors created memos in NVivo after each interview to express their ideas, thoughts, and insights related to an emergent code or category. This process is referred to as theoretical memoing, and it is crucial in ensuring quality in GT studies as it enables the authors to enhance relationships between codes.
- iv. *Sorting.* Once data collection was finished, the authors started the sorting process. This process is useful in explaining each category and relationships in detail, consequently, formulating the theory grounded in data. It is noteworthy that the low number of interviews limits the generalizability of the emerging theory. In order to minimize this threat, the authors triangulated their findings with a minimal literature review which was performed at this stage of the process.
- v. *Theoretical coding.* Theoretical coding is aimed to discover relationships between the core category and the related categories and formulate the hypotheses that explain the emergent theory. The authors identified the relationships between the core category “BC implementation in organizations” and the other related categories: *key activities*, *challenges*, and *success factors*, as presented in Figure 2. Glaser proposed a set of theories' structures, referred to as theoretical coding family.⁴⁰ In this study, the authors adopted the temporal/process coding family to explain the emerging theory, as illustrated in Figure 3. This coding family presents a category in terms of stages, timeliness, conditions, phases, actions, and temporal ordering of work.⁴⁹

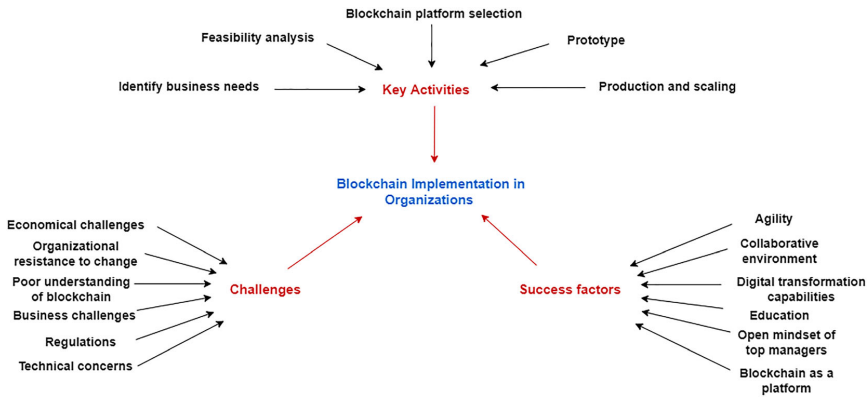


FIGURE 2 Emergence of grounded theory from the identification of categories and concepts.

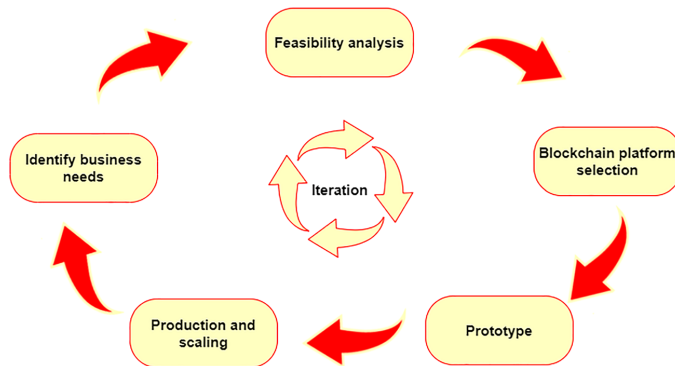


FIGURE 3 Presentation of the “key activities” category in the form of Glaser’s temporal/process family

4 | RESULTS

The emergent GT in this study encompasses one core category *BC Implementation in Organizations* and three related categories: *key activities*, *challenges*, and *success factors* (see Figure 2). These categories are explained in the following sections.

4.1 | Key activities

Data analysis indicated the emergence of key activities in the implementation process of BC technology in organizational settings. These key activities are depicted in Figure 3 in the form of Glaser's temporal/process family.

4.1.1 | Identify business needs

The first step of implementing BC into an organization's environment is to identify business needs. According to EX2, the main factor in any successful implementation is to define the scope, requirements, and then find the best tool to address these requirements. For instance: "the typical requirements for supply chain would be traceability. So, you have a product and you want to know who has touched it along the way, where did it origin from, has it met ethical standards throughout the value chain, has it met environmental standards ... and then decide what technology you can trust to ensure traceability" (EX2). This expert strongly supported that requirements should drive the technological choice and not vice versa. Likewise, EX3 emphasized that organizations should not decide to implement BC because it is a hyped technology with intrinsic potential, but because this technology solves a problem that the organization may have. "It is not BC because it is BC, but because it solves a problem that they may have" (EX3). Likewise, previous studies have reported on the perception of BC as a solution looking for a problem.⁹

Another important aspect to consider in this phase is that BC does not fit for every use case, as agreed by our experts. In fact, BC use cases are very specific and constrained (EX6, EX7). In spite of this, the experts considered these use cases as not difficult, but they claimed that practitioners are not trained to recognize them (EX7).

4.1.2 | Feasibility analysis

After identifying the business needs and use case, it is essential to analyze if BC is the most suitable solution to address the identified business needs compared to existing alternatives. EX7 revealed that the process of investigating the right environment for the implementation of BC takes 60% of the whole implementation process. This expert suggested to consider two main dimensions in this regard:

- i. *The legal dimension*. Do you need to act with other organizations or clients that may not agree with what you claim? EX7 explained this dimension further: "when you put BC in a context, it is because you need to be able to demonstrate that data is right to people who may not agree with what happened before ... You need to make producers and consumers accept the data they have installed, so you need to put it in a legal context that they will accept to avoid going to trial." If this does not exist, then a shared database would be the most appropriate solution.
- ii. *BC as a platform*. Do you have a market you want to join or do you intend to create an ecosystem for others to join? EX7 further stated "If you are not talking about a selling platform or to be or to put business in, you do not really need a BC. You have already the cloud for that." After identifying the need for BC in a specific organizational environment, it is important to understand the value and benefits of applying such a technology. EX6 provided an example in which BC was used to prevent double spending of the city pass vouchers and to keep track of who was using vouchers and how. Although EX6 acknowledged that this could be done with existing technologies, the expert pointed out the following benefits of using BC in this context: transparency, cost-effectiveness, and reliability. However, as concluded by the experts, benefits such as transparency and reliability might be difficult to capture and require new measurement methods.

Furthermore, the organization has to make a set of strategic considerations, for instance, where the computers will be, how to ensure their security, and how will they communicate with each other. Afterwards, the process proceeds with the identification of the number of nodes of each company, the types of tokens to be signed, and the acceptance or rejection rules for the different tokens. It is noteworthy that not every computer on the network can be considered suitable to sign specific transactions. For instance, transactions committed by providers should be checked by institutions related to the providing process, but not by providers that are related to the final consortium (EX7).

Furthermore, as pointed out by EX7, "BC is only one part of the solution you are putting into market, but it is not the whole solution." While it is true that BC is responsible for protecting the stored data and transactions, it is also true that BC is not responsible for what happens before

and after. Therefore, EX7 outlined that “we need to focus on how to link the systems that go before the BC [sensors, machines, any kind of hardware] and systems that go after the BC [artificial intelligence, ERPs, any kind of software].” The last concern raised by the experts entails the delay of certification that the whole network is performing about transactions and data stored. In this regard, it is necessary to make sure that this delay is acceptable for the whole context.

4.1.3 | BC platform selection

The experts provided valuable insights into the BC platform selection process. For instance, EX8 selected the most suitable BC platforms for specific use cases in the banking domain and elaborated on the reasons behind these choices: “We decided to use IOTA because we needed an Internet of Things technology and you cannot use Ethereum, Bitcoin or Hyperledger because it is too big. We used Ethereum when it comes to buy and sell shares on the stock exchange and the reason was programmability and that the smart contracts generation is very efficient. But you cannot use Ethereum when it comes to money because you do not know who is paying and who is receiving. So, we needed to use Corda which is a private chain”

According to EX4, foundational values of the organization should drive the underlying technological choice: “If you support the anarchistic values of Bitcoin, in a way to disrupt banking and maybe governments, then this is something you should build into ... If the energy consumption is a problem for you, then you should look into a technology that uses less energy. So, I guess you should know what your values are, and from there choose a platform.” According to this expert, the alignment between the organizational values and BC platforms is more important than quantitative analysis, for example, the number of transactions per second, given the emergence of new platforms and solutions such as Ethereum Layer 2 scaling solutions (EX1).

Other factors to be considered when choosing the BC platform are as follows:

- i. Network accessibility which entails the selection between public and private BC platforms: “If you need a public platform, you are saying that you need the crowd saying that what you are doing is right. If you need a private platform, you are saying that you need some actors or companies to say that what you are doing is right” (EX7).
- ii. Transaction fees. As EX9 stated, “transaction fees in platforms such as Ethereum are very high at the moment. So, you usually end up with an application that costs a lot of money to run. New platforms such as Avalanche and Cardano have supposedly lower transaction fees.”
- iii. Consensus mechanism was mentioned by two of the experts (EX1 and EX4). EX1 elaborated on this factor: “it has been recognized that proof-of-work which is the consensus mechanism of traditional BCs like Bitcoin is very energy-intensive. And that has become controversial. So, it appeals for new types of consensus. One of them is proof-of-stake. In IOTA we have developed something that is even an alternative to proof-of-stake, so it is a customized version of that.”
- iv. Programmability. According to EX3, it is important to consider whether the BC platform supports SC and the programming language used to build SC.
- v. Community of developers. Two of the experts (EX8 and EX9) suggested considering the developers' community when selecting the best fitting BC platform. EX8 outlined the importance of this factor: “what is good with Ethereum is that if I would like to make a new project, and I am looking for a programmer, then there are hundreds or thousands of people doing Ethereum and it is very important that there is a market.”

The majority of the experts confirmed that organizations face difficulties when selecting the best fitting BC platform. Two of the experts (EX3 and EX9) explained the difficulty with the fast-moving pace of BC technologies and their expansion in various domains. New platforms are emerging and as admitted by EX9: “it is difficult to follow on everything.” The rapid increase of the number of BC platforms in the market has been identified as a significant challenge for organizations also in literature.⁵⁰ EX1 claimed that currently there are no standards to guide the selection process. The expert also stated that “there is a lack of clarity in the BC space, so you have lots of BCs pretending to be doing something, but in the end they do not perform as expected or do not scale towards what they claim.” According to EX1, misleading indicators such as the performance of cryptocurrencies may contribute to the lack of clarity because “there is a whole domain of BC that is not reflected into these cryptocurrency trends” (EX1). Conversely, EX4 recommended the pragmatic approach of “following the stream.” According to this expert, monitoring and following what BC platforms other developers, organizations, and governments are choosing is a safe and cost-effective approach. “If there is a lot of development happening, if the tooling is good, if the security audits are happening all in one BC, then you get a lot of stuff for free. But if you were to use a lower ranked BC, then you will have to use your own resources for creating development tools, testing tools, security audits”

Furthermore, EX3 explained the difficulty in selecting BC platforms with the dependence of SC language with the underlying BC ledger. This expert pointed out that existing solutions are built with one SC language, which is bound to a specific BC platform. Consequently, “if the organization wants to then shift [into a new BC platform], a lot of rework is needed to be done” (EX3). Therefore, EX3 emphasized the importance of SC

that are able to talk to different ledgers because “that makes the choice easier for decision-makers in the sense that they can build a solution using one smart contract language and they would be assured that the solution that they built can work, irrespectively of the ledger they may choose in the future” (EX3).

4.1.4 | Prototype

The experts outlined that based on their previous experience with various companies, implementing BC is rarely a decisive choice of the management. Instead, “it is more of a slow testing prototype thing, often through the innovation people or the innovation department. They test out a prototype and then, they move it slowly into the real organization, into production, which almost never happens” (EX4). EX7 refers to this process as the pilot stage, which entails creating a simulated network that covers the whole process to test whether the process is right and can work. Dozier and Montgomery⁹ concluded that organizations test BC through prototypes and proof of concepts to maximize the comprehension, minimize risks, and identify the business value of this technology. Furthermore, the experts recommended carrying out a performance analysis of the BC-based solution in terms of latency, transaction throughput, and transaction speed (EX8, EX9), security analysis when dealing with sensitive data (EX8, EX9), and user-interaction analysis to evaluate how the user interacts with BC (EX9).

While BC applications have become more available and easier to develop with time (EX3, EX4), three of the experts (EX1, EX3, EX7) agreed that the most difficult part of the implementation process is designing the right BC-oriented solution. EX1 claimed that “a lot of the bottlenecks are in the development phase, rather than scaling or implementation. Developing the solution means to design it. A lot of people have a problem understanding BC and how to design a solution that utilizes it.” Likewise, EX4 confirmed the difficulty of the design phase: “I think having the right design for the client is the hardest part because our minds are not really thinking the BC way. We are thinking the services way, we are thinking that we have fast service on the cloud, that we can query and get answers. In the case of BC, you can query but to get answers you need to accomplish some rules and these rules must be consensual among all parties having the infrastructure”

4.1.5 | Production and scaling

The pilot stage is followed by the production and scaling stage that aims to support the increasing volume of transactions and data. Previous studies have reported on limited production implementations of BC systems^{6,9} and scarce empirical studies investigating the reasons.⁶ The experts confirmed that most of the projects stop at the pilot stage and provided three main reasons. According to EX6, this may occur because the solution did not deliver the expected value. On the other hand, EX4 perceived this as a strategic problem rather than a technical problem. Furthermore, EX4 elaborated on this perception: “I guess the bottleneck in organization is often that the receivers or the people working in production, or with real life products when they are faced with taking over the project from the innovation team, moving from prototyping into production show a lot of resistance, because maybe they hadn't been included or maybe they hadn't included themselves in the process” (EX4). The organizational resistance to change barrier has been explained in detail in Section 4.2.2.

Finally, EX3 who is specialized in building prototypes and proof of concepts for different customers considered the scaling of these solutions challenging: “We provide MVPs, proofs of concepts for different customers, but then when the scaling of the solutions does come forward, it is much harder because it goes back to this business discussion so it starts to involve other parties, and then the doubts from these different parties starting being raised on who controls what, what are the benefits for my side to actually onboard such system.” According to EX3, one of the easiest ways of putting a BC system into a full-scale operation is when there is one central party that has a lot of market power to push the other parties into the system.

4.2 | Challenges

4.2.1 | Economic challenges

All the experts discussed the costs related to the implementation of BC in organizations. As EX7 pointed out, “having a decentralized network with different machines, working in different places that need to be protected and secured is not the cheapest thing you can think.”

The experts mentioned five types of costs: (i) transaction fees, which are significant when building on public BC platforms (EX1, EX3, and EX9). EX1 referred to these fees as “not only substantial, but also unpredictable”; (ii) legal costs “when you try to live in that gray area, in the brink of something illegal, then you have to use a lot of money on lawyers making sure you do not end up in jail” (EX4); (iii) cost of shifting into another BC platform (EX3, EX4); and (iv) development and infrastructure costs (EX7) which EX7 categorized into three groups:

- i. *Application for providers.* These costs vary depending on whether you have to update existing software, find a standard application that can work or develop a new application for providers. EX7 outlined challenges faced in each of the aforementioned options: “there is no standard application, so you have to pick different solutions,” “No one wants to expand their applications to let them work with BC, because no one wants to pay,” “the problem is that you have a lot of companies having their own vision on what should be done. In the end, you need to make everyone happy and this is not easy.”
- ii. *Applications for consumers.* Consumers are not always the same entities as providers; hence, they do not use the same applications. In fact, consumers may vary a lot. EX7 illustrated the difference between consumers and producers with the following example: “producers in a market for selling shoes, they all make shoes. So, their software can be similar or maybe they use a standard software ... And consumers could be stores, parties in the middle, consumers could be the same providers. So, the ecosystem of applications you have may vary more.” The most cost-effective solution in this scenario would be to create the same application for providers and consumers. Otherwise, the cost of each application may vary from \$20,000 to \$100,000–\$300,000 depending on the complexity of the application, according to EX7.
- iii. *Infrastructure costs entail data storage, network communication, and hardware.* The *data storage* depends on the type of data transferred and the amount of data. However, it is noteworthy that storage costs are much lower compared with those from 20 years ago (EX7). Regarding *communication*, there is a lot of information that moves from providers to the network, inside the network to reach synchronicity of machines, and information going out. Therefore, it is important to determine the number of clients and providers and enable machines to avoid data movement redundancy. Finally, it is important to determine *hardware costs* that depend on the number of nodes.

EX7 presented the whole picture of costs involved in BC projects: “If we are talking about a small network of 3–4 nodes having one kind of provider, I do not really care if they are 3 or 5 companies but they act if they were just one, they use the same software and they are putting the same kind of data inside. And when it happens something in the same way with the clients, then you can start pricing the whole solution, maybe \$250,000. But when you have a complex scenario, when you have a lot of companies, a lot of providers, a lot of consumers, prices go really high compared to that.” Indeed, financial stability of the organization is necessary in order to support the implementation of BC.⁵¹ Therefore, our results suggested that the decision on whether to implement BC given the associated costs also depends on the economic environment of the organization. As stated by EX8, “if an organization has a lot of money, then there is tolerance ... if there is not so much money, the organization is cautious about what to implement.”

4.2.2 | Organizational resistance to change

The experts considered the organizational resistance to change as a core challenge when implementing disruptive technologies such as BC. This is in line with previous research on the topic.⁵² Walsh et al.⁵² attributed the resistance to the lack of standards and regulatory backing. Our results suggested that the resistance to change and skepticism is related to the hype of the technology: “people will be against it [BC], even if it would make some sense, but they would be for it even if it would not make a lot of sense to implement it” (EX6). Moreover, three main factors that contribute to organizational resistance to change emerged from data analysis:

- i. *Innovation–production gap.* As EX4 outlined “BC is a technological choice, which is pushed through by innovation people who are often business people.” The pressure to implement BC solutions does not come from the top, but from the innovation department. Therefore, technicians working in production or with real-life products are resistant when they take over the project from the innovation team. EX4 explained that this may occur due to their lack of participation in the process (they have not been included or they have not included themselves) or their unwillingness to utilize this technology.
- ii. *Conservative management.* EX1 raised the concern of conservative managers or decision makers that challenges the implementation of BC in organizations: “they are thinking operationally how to maximize the performance of the organization and when you bring BC you need to divert from the usual course of action.” Two of the experts (EX8 and EX9) referred to specific domains that are conservative: healthcare and banking. EX9 raised the following concern: “to make changes in the healthcare industry takes a lot of time and I think a lot of people in the decision-making positions are quite reluctant to disruptive changes and if you look at electronic health records, they are controlled by a few very large actors that supply these systems and they have not adopted blockchain.” EX8 stated that banks are slow in adopting new technologies, especially when it comes to cryptocurrencies: “they do not like cryptocurrencies and the reason is financial authority. They have to know who is paying and who is receiving ... However, when it comes to using BC as a technology for sharing information, they are very much looking into it, at least in Norway.”
- iii. *Centralized mentality.* EX3 reported on having worked with different entities with a centralized mentality. According to EX3, “this is their *modus operandi*, but it defies the purpose of using distributed ledger technology or BC, if an entity wants to control everything.” In order to change this mentality, education is essential.

4.2.3 | Poor understanding of BC

All experts acknowledged the issue of poor understanding of BC. This challenge has been also confirmed in the existing literature.^{6,51–53} EX1 reported on the confusion between the concepts of SC and BC and the tendency to overuse SC in any context. In this regard, EX1 clarified: “you can use BC without an association with SCs and smart contracts is just a layer on top of it” In addition, EX1 outlined the difficulty in designing solutions that utilize BC due to the poor understanding of this technology. On the other hand, EX3 acknowledged that BC solutions have become more available and easier to develop, but there is a gap from most companies in understanding the potential. In the same line, EX10 reported that decision makers have difficulties in understanding the technology and its potential: “they think the implementation of BC can be similar to implementing CRM [customer relationship management] or ERP [enterprise resource planning].” These difficulties have been confirmed also by EX9: “a lot of people have trouble wrapping their head around the decentralized governance part of the technology since this is quite new.” Four of the experts (EX2, EX3, EX4, and EX8) asserted that BC is immature and companies fail to understand the appropriate application of such a technology.

On the other hand, EX6 perceived the hype around the technology as the problem, rather than the poor knowledge: “people have polarizing views on how beneficial it [BC] is, so they either are completely against it without knowing much about it, or they are strongly supporting it even though they know some of the drawbacks and some of the use cases they are pushing will not be possible to implement.” This expert strongly believed that the hype that has been raised around this technology will never be matched regardless of the improved BC platforms that will emerge in the future.

4.2.4 | Business challenges

EX7 considered BC a “business technology,” which is pushed into an organization’s environment by business people who identify the need for new markets: “A technician will never think of BC. The technician will think of the cloud that (s)he can manage and can make it grow in an easy way and have a lot of techniques that are really prepared to do that. When you put BC inside companies is because the business part of the company says, we need new markets. We need to create incomes from collateral activities, and we need to join the rest of our clients.” However, experience has revealed that business people are not committed to understanding the potential of BC (EX7). This expert described how subject matter experts assist business people in understanding BC “When we go to a new company that is interested in BC, we say to them, we do not want to talk with your tech people. We need to talk to business people and tell them you are losing clients because your clients are going to that market that they have created. Well, you need to join that, but you need to understand first, what the internet of value is, and how you can use it in your organization to grow your business and to create new businesses around. So, it is 50% working with business people, 50% working with legal people. When everything is fine, tech people enter the equation.”

The implementation of BC-based solutions requires the willingness of organizations to take a step forward and be ready to digitally transform their businesses into new business models (EX3). According to EX1, “BC-based solutions are not just plugins to enhance cost efficiency. Most of the time, they bring some sort of business model transformation.” Organizations need to make strategic considerations either by moving into a new market or by creating new products from their old products. EX4 illustrated this with an example: “you cannot do lending as you do today, but you will have to look into decentralized finance ... If you are utilizing kind of ecosystem building, opening up your data, that would mean that your product will be something new, but you will still do lending in a way. And that could of course change how your whole business is built.”

4.2.5 | Regulations

While it is true that regulations may reduce the level of experimentation because of associated risks, it is also true that regulations ensure that what gets experimented with is well-thought and less risky (EX3). Hence, EX1 outlined the importance of being aware of regulations in the market in which the organization operates and being willing to engage in regulatory discussions. This expert has dealt with different regulators in various industries and revealed that these parties set up sandboxes to allow innovation to get observed and understood. Therefore, the organization that aims to implement BC needs to be ready to be exposed to this type of environment. Interestingly, EX1 encouraged those who believe in the potential of the technology to stand in front and push to the reshaping of regulatory frameworks. However, EX3 admitted that working with public BCs is unlikely to happen in a realistic way, at least for regulated industries, due to the need to be approved by regulators. Some entities may, however, sidestep regulations and experiment anyway, given that public BCs offer this capability.

On the other hand, EX4 and EX9 believed that the problem is not regulation per se. EX9 considered the fear of regulation and confusion related to whether the BC is aligned with current regulations as the crucial issues: “mainly when I talk about this [BC] technology with people within the health domain, I usually get that question. Can we implement this with the current regulation and I do not see any reason why not ...

There are gray areas, but there is no sort of regulation that really put sort of a break towards it. But I think, the fear of regulation and maybe a little bit of confusion if it is suitable with the current regulation or not.”

In addition, EX4 raised concerns about ethical issues introduced by entities who try to bend regulations: “It is very challenging because what these companies will try to do is often in a gray area of the law or regulation and that is not the fault of the regulation, it is because people will try to bend the rules because it is the BC.” Furthermore, EX8 acknowledged the importance of regulations but claimed that regulators are slow: “If the regulators were faster and they could regulate the market, then you could trust the market and more money comes into the market.” Likewise, EX7 argued that technology moves to a faster pace compared with regulations which means that what is being regulated risks to become outdated: “they are trying to regulate the picture of how Bitcoin and Ethereum were two years ago. That is so far from where we are now.” This expert introduced another problem which entails the need for an invitation to join BC ecosystems. Once an entity joins, the BC assures that all the data and transactions are protected mathematically and shared among all entities. However, BC does not take responsibility for what happens outside the platform. As EX7 stated: “this makes BC a private club ... We have to require that the government is invited to every private network and why should they be”

One of the most critical problems discussed by the experts is the General Data Protection Regulation (GDPR). The immutability nature of BC goes against the right to be forgotten enforced by GDPR. EX4 elaborated on this issue: “you will need to always put one part of private data on BC, which can be the Ethereum address or Bitcoin address which is not erasable ... But it is a challenge to store private data on BC forever, which goes directly against GDPR.” In addition, EX6 reflected upon storing data on BC in relation to GDPR: “according to GDPR, you are only allowed to gather what you specifically need or must have. BC of course collects a lot more usually” The absence of compulsion to comply with data privacy regulations such as GDPR has been considered a barrier also by previous studies.^{3,54}

4.2.6 | Technical concerns

Interoperability is an important concern that has captured the attention of BC researchers.⁵⁵ Organizations may choose different BCs that do not communicate with each other. This defies BC’s purpose of sharing information by creating additional silos. EX3 explained this technical limitation, as follows: “interoperability is definitely a concern that may impact the implementation of such systems, because as people start to see that different [BC] platforms are popping up, and they decided to use one instead of the other. So, different organizations may see that it ends up being similar to centralized systems, because different organizations are in different systems, so siloed and these silos do not communicate with each other” EX3 outlined the need for SC languages that are able to talk to various ledgers. The idea is to build solutions that work, irrespectively of the ledger that organizations choose in the future. Furthermore, this expert elaborated on this issue: “right now, if you choose to go with Ethereum, it bounds to solidity. If you go with Hyperledger Fabric, you are bound to the chain code. If you are going for R3 Corda, SCs are developed through Java and Kotlin ... If for instance, you have a SC language that can talk to all these, you kind of separate this aspect from the underlying ledger.” Currently, EX3 is working with a technology that enables an SC language that is independent of the underlying ledger. EX2 emphasized the importance of abstraction and encouraged discussions regarding abstraction rather than BC itself. This expert quoted one of the customers who is the head of US Homeland Security’s Silicon Valley Innovation Program: “We want to stop talking about BC ... We want to abstract that so far away that we never need to talk about it at all.” Currently, EX2 is working with decentralized identifiers and verifiable credentials as an abstraction that sits on top of distributed ledger technologies.

Furthermore, the experts mentioned other technical concerns. One of the concerns is scalability, which is the capability to process transactions to the same degree of efficiency if the network grows and the number of transactions increases (EX1). Another concern is security, that is, organizations need to host BC systems and guarantee that they are secure against attacks that may deny the service and are inaccessible by unauthorized users (EX3). Security issues are even more severe in safety-critical domains. EX6 claimed that “some of the blockchain applications have not been very good at that [ensuring security in safety-critical domains].” This expert supported this claim with several attacks against cryptocurrencies that have led to significant financial losses. On the other hand, EX7 perceived BC as “the best technology we have created thinking about security” and supported this perception by claiming that “in 10 years working with Bitcoin, no one could break it, and we are talking about something that a lot of people would like to break to pick the money. So, it makes it really stable, and really secure.” Paradoxically, this expert considered BC as “the most vulnerable technology we have put into market” because “all computers have the whole storage.” EX7 elaborated on this concern: “your network is as secure as the least secure computer on the network, because if I am a hacker, and I can enter one node, I can steal the whole information on the system. So, instead of having one computer in the cloud that I have to attack, I have thousands of machines that I can attack. So, we have created a lot of points of attack for hackers.” This is in line with other studies that identified security as both a benefit and a challenge of BC systems.³ Although the increased security built into the design of BC is a benefit for enabling peer-to-peer transactions, several attacks such as account-take over, 51% attack, hacking, and digital identity theft are theoretically possible.³ However, Bitcoin, the first BC application, has not been successfully attacked.²

4.3 | Success factors

4.3.1 | Agility

In order to implement BC successfully, organizations should be agile, meaning that they need to be ready to experiment in an iterative fashion and to learn from mistakes quickly (EX1). Likewise, EX5 encouraged small iterations in order to understand problems quickly and prioritize them. According to this expert, “one of the biggest mistakes organizations are doing is that they start out big, they do it too fast, and then they do not understand what they are doing.” According to EX1, many practitioners are not knowledgeable in this regard; therefore, this expert recommended “setting up dedicated teams working as a task force to actually do these developments, rather than starting from the existing teams and operations.” Moreover, EX3 stressed the importance of close collaboration between subject matter experts or startups that offer their BC expertise and customers by involving customers in all the meetings and sessions that are carried out internally. In this way, customers can be aware of the different tasks that need to be developed.

4.3.2 | Collaborative environment

BC aims to break the silos between different types of organizations. A crucial requirement to fully leverage this technology is the collaboration not only on an intra-organizational level but also on an inter-organizational level.⁸ As EX1 stated, “It is not enough to apply it downwards into your own organization; it is about how you collaborate with the other parties, so you need to have collaborative environment.” Likewise, EX5 claimed that the main benefits of using BC do not come by just applying it in your organization, but by creating the “network effect.” Frizzo-Barker et al.³ define the network effect as the concept that the value of a product or service is based on the total number of users. In addition, EX1 highlighted the importance of creating new value streams in collaboration with other parties and referred to this as innovation ecosystems. This collaboration can take the form of alliance structures in which all the parties can define together how the solution will be orchestrated and governed by the different parties (EX1). Moreover, collaborations can also be in the format of public-private partnerships. EX1 illustrated this with the IOTA case: “IOTA technology can be utilized for public sector level, as part of the digital infrastructure and you will have the private sector on top, leveraging that technology and using it.”

Furthermore, EX4 pointed out that collaboration is essential in a BC ecosystem; however, it has taken a new decoupled form. This means that there is no need to know the other parties or have bilateral agreements; instead, you just need to integrate with them and leverage their SC or services in your service. Finally, EX9 mentioned the importance of collaboration between industry and academia. According to this expert, most of the developments within the BC sphere are carried out by startups in the sector. Although the development is faster in the private sector than in the academic sector, EX9 reported that in the healthcare domain, executives are very interested in academic research to support their decision to implement BC.

4.3.3 | Digital transformation capabilities

The experts had different views on the importance of digital capabilities of organizations that intend to implement BC. EX1 raised the concern that “companies that are not techie are going to struggle a bit” and suggested companies “to be ready to hire some talent, probably people that are of younger generation who know how to manipulate those technologies.” Similarly, EX3 acknowledged the importance of hiring talents in order to bring a different vision, but outlined the need to be balanced with the business knowledge that experienced people have. However, three of the experts (EX3, EX5, and EX6) did not perceive this factor as critical for most organizations, given the possibility to leverage BC startups and consultants. Despite leveraged expertise, it is still important for an organization to understand what can be done with BC and what value it can bring. In addition, EX7 strongly recommended organizations to invest in training existing personnel and getting BC knowledge in-house, due to the scarce market of BC experts, which has also been confirmed by other studies.^{3,53}

4.3.4 | Education

The experts confirmed the need for education in three perspectives: (i) education on how to recognize BC use cases because “BC is not a solve everything solution” (EX4) or “BC does not fit for everything” (EX7). According to EX6, many people treat BC as a *silver bullet* that enables you to do a variety of things. In fact, there are very specific and constrained use cases that may yield benefits from using BC, (ii) to educate managers regarding the business value of implementing such a technology (EX5) and to change their centralized mindset (EX3, EX9); (iii) education from a technical perspective in order to develop expertise on how to build BC-based systems (EX3). However, understanding what knowledge is required

to implement BC in organizational settings is not trivial because “BC is not a technology that was created in a university environment or laboratory ... So we cannot say this is the standard and this is what everyone should know to put the technology into real work ... Universities try to do that but technology evolves in an uncontrolled way” (EX7).

4.3.5 | Open mindset of top managers

One of the key success factors of implementing BC in organizational settings is to approach it with an open mindset as innovators and be willing to take some degree of risk, which is inherently associated with innovation (EX1). EX1 outlined the importance of being open to change: “When you bring BC, you need to divert from the usual course of action.” In the same line, EX4 considered BC a new way of doing things, and therefore advocated “openness to change and new way of thinking.”

4.3.6 | BC as a platform

The experts suggested interested organizations to think of BC as a platform. EX7 encouraged organizations to consider implementing BC if there is a market they want to join or if they intend to create a market for others to join (see Section 4.1.2). EX4 reported on previous communications with different organizations and discovered that “they want to leverage data in a way that benefits them and not their competitors. However, that does not make much sense in the BC world. It is important to create the platform which other parties are using and thereby giving away a lot of power and data for free to get people into your platform, to get people to build on top of your data, to leverage your ecosystem.” EX3 revealed that getting new organizations or entities into your platform is not a trivial process: “When you want to invite new organizations to join your consortium or join your efforts to build a system where you interact with multiple parties on a certain topic, that is where the conversations just keep going and going. Agreements are not always easy and quick.”

In addition, according to EX4, it is important to think big, and not try to digitize only a part of a process or put it on BC because it will not make sense from the business perspective. In such a case, existing technologies would be more suitable. The efficiency of BC can be perceived in the case of a larger chunk of the process that includes many parties. EX4 illustrated this with an example: “working with the Norwegian Business Registry, I see that the value for them is not kind of tokenizing shares and just creating a digital representation of the company’s stocks, but it is building the ecosystem so when they deliver the foundation which is a tokenized stock, then any company can come in and expand on that revenue stream with functionality and businesses, so that could be crowdfunding, lending, it could be new kind of portfolio systems either for viewing or for doing index funds.”

The main findings are summarized in Table 3.

5 | DISCUSSION

5.1 | Results evaluation

Glaser⁴⁰ introduced four criteria to evaluate the validity of the emergent categories that compose the theory: fit, workability, relevance, and modifiability. These criteria are explained in the following section:

- i. *Fit* refers to codes, categories, and a theory that emerge from data, rather than preconceived perceptions, views, and biases of researchers. The authors of this study did not perform an extensive literature review in the initial phases to prevent preconceived perceptions, views, and biases from influencing the data. In fact, data only emerged from the interaction between the researchers and participants. In addition, the authors selected a diverse set of participants in terms of job positions, project domains, and countries (see Table 2) to ensure different perspectives on the emerged concepts and categories and fulfill the *fit* criterion.
- ii. *Workability* refers to the extent the core category is integrated with related categories and the extent the theory explains the area under study.
- iii. *Relevance* refers to the extent the theory enables the emergence of core problems and processes in the area under study. In order to assess the *workability* and *relevance* criteria, the authors provided participants with a preliminary report of the results. BC experts were asked to provide feedback on the results. In what follows, the authors present the feedback provided by a few experts: “I do think results provided are valid and relevant,” “There is always a bias in all implementations, but I can figure out real situations behind the rationale depicted. Maybe it is not covering all cases, also true” “some of the most important challenges can be derived from results provided. It paves the way towards better and clearer challenges, and opportunities can be seen there,” “very interesting results.”

TABLE 3 Summary of the main findings

Categories	Main findings
Key activities	<ul style="list-style-type: none"> Requirements should drive the technological choice, not vice versa. BC does not fit for every use case.
Feasibility analysis	<ul style="list-style-type: none"> The process of investigating the right environment for BC implementation takes 60% of the whole implementation process. Organizations should consider two dimensions that influence the decision on whether to use BC: the legal dimension and BC as a platform.
BC platform selection	<ul style="list-style-type: none"> Foundational values should drive the BC platform selection process. The following factors should be considered when selecting the suitable BC platform: network accessibility, transaction fees, consensus mechanisms, programmability, and community of developers.
Prototype	<ul style="list-style-type: none"> The most difficult part of the implementation process is designing the right BC-based solution. The decision to implement BC comes from the innovation department because of testing out a prototype, rather than the management. Most BC projects stop at the pilot stage.
Production and scaling	<ul style="list-style-type: none"> Findings reveal the following costs associated with implementation of BC projects: transaction fees, legal costs, costs of shifting into another BC platform, development and infrastructure costs (costs for providers' applications, consumers' applications, data storage, network communication, and hardware). Three main factors contribute to organizational resistance to change: (i) innovation–production gap (ii) conservative management (iii) centralized mentality.
Economic challenges	<ul style="list-style-type: none"> Decision makers have difficulties in understanding BC and its potential. BC is a business technology that is pushed into organizations by business people. Technology moves at a faster pace than regulations. Absence of compulsion to comply with GDPR The main issue is not regulation per se, but the fear of regulation and confusion whether BC is aligned with current regulations.
Organizational resistance to change	<ul style="list-style-type: none"> Main technical issues to be considered are interoperability, scalability, and security. Lack of interoperability leads to the creation of silos that do not communicate with each other, like centralized systems. The need for smart contract languages that are independent of the underlying ledger. BC can be considered the best technology created considering security, but at the same time the most vulnerable technology.
Poor understanding of BC	<ul style="list-style-type: none"> Organizations should be agile, use small iterations to identify problems quickly and prioritize them.
Business challenges	<ul style="list-style-type: none"> Close collaboration between BC experts and customers
Regulations	<ul style="list-style-type: none"> Importance of establishing innovation ecosystems in collaboration with other parties to create new value streams
Technical concerns	
Agility	
Success factors	
Collaborative environment	

TABLE 3 (Continued)

Categories	Main findings
Digital transformation capabilities	<ul style="list-style-type: none"> • Collaboration can take the form of alliance structures or public-private partnerships. • In the BC ecosystem, collaboration has taken a new decoupled form. • Most of developments are carried out by BC startups. • Development within the BC sphere is faster in the private sector than in academic settings.
Education	<ul style="list-style-type: none"> • Organizations should not rely solely on BC startups and consultants, but train existing personnel and get BC knowledge in-house. • There is a need for education in three dimensions: (i) on how to recognize BC use cases (ii) educate managers regarding the business value of implementing BC (iii) technical education on how to develop BC-based solutions.
Open mindset of top managers	<ul style="list-style-type: none"> • Approach the implementation process of BC with an open mindset, openness to change and risk, and new ways of thinking.
BC as a platform	<ul style="list-style-type: none"> • Think big and do not digitize only a part of the process. • The efficiency of BC can be perceived when digitizing a larger chunk of the process that involves many parties.

Abbreviation: BC, blockchain.

Based on the feedback received, the authors confirm that the emergent categories work because they explain the reality of the area under study and are relevant because they allow core problems and processes in the BC sphere to emerge.

- iv. *Modifiability* refers to the ability of the theory for continual modification and development with the occurrence of new data. The categories that emerged in this study went through different modifications over time due to the constant comparison techniques that were followed. The authors compared new data with existing codes, concepts, and categories until the theoretical saturation point was identified. Because this process allows the results to be modifiable, future work may focus on evolving the categories upon receipt of new and more diverse data.

5.2 | Implementing BC for RT using the emerged concepts

In this section, the authors aim to improve the initial BC-enabled RT framework presented in their previous study¹² using the emerged concepts (see Section 4). Such an initial framework consisted of four elements: strategic layer, BC proposal, implementation, and assessment. The improved framework (depicted in Figure 4) keeps the BC proposal at the core and integrates the other three elements into the core categories as follows.

5.2.1 | Identify the need for distributed RT

As recommended by the experts, the first phase should commence with determining the scope, identifying business needs and requirements, and then choosing the best tool to address these requirements. The focus of our case study is on RT in interorganizational software projects. First, it is important for organizations to identify the need for RT in such projects. Rempel et al.³² provided a good overview of the importance of RT in distributed software projects, as a proof-of-quality and correctness of the end product to avoid costly disputes. In addition, traceability is needed to track the project's progress to determine if the project can finish in time, within budget and with the expected output, and to verify the compliance of the project execution with legal regulation. In spite of the importance of sharing traceability information among participating entities, previous studies revealed that this information is not always disclosed and relied upon due to organizational boundaries and conflicting objectives.³² For instance, the organizational goal of one project partner may be to keep technical knowledge confidential, and this may contradict the need of

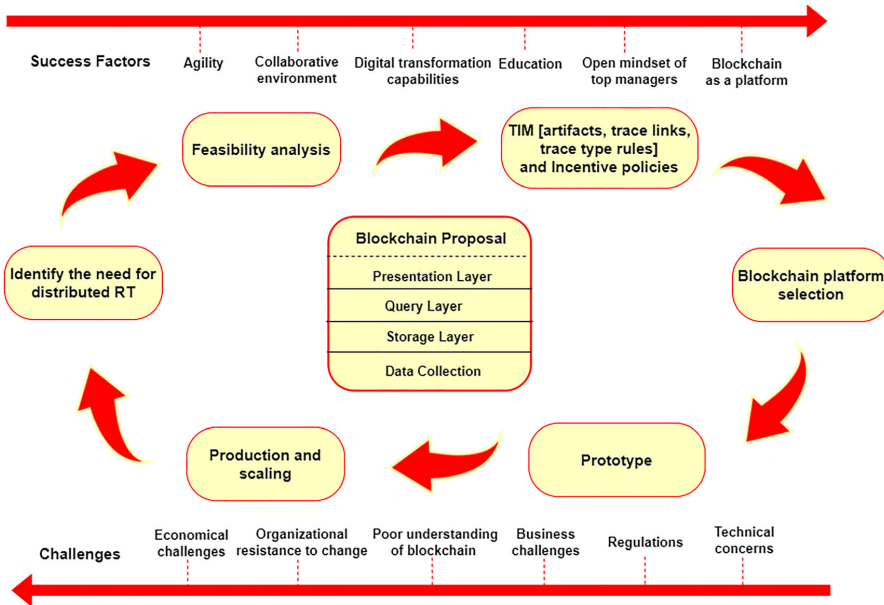


FIGURE 4 An implementation framework for BC-enabled RT

the other entity to gain product knowledge, which could be used for further release planning.³² These problems enable us to deduce the following requirements for traceability information: available, shared, reliable, and trusted. Organizations should identify a suitable solution that addresses these requirements and the ultimate scope of enabling a shared, reliable, and available traceability knowledge base that ensures visibility to all the participating entities regarding what/how/when trace links were created and by whom.

5.2.2 | Feasibility analysis

In this phase, organizations should decide on the most suitable technology to address the identified needs. BC can be a viable technical solution to ensure reliability and availability of traceability information in interorganizational software projects.²³ This technology needs to be compared with existing solutions for the storage and representation of trace links such as cross-reference tables, traceability matrices, relational databases, and graph traceability repositories.⁵⁶ Our results suggest that the assessment of whether BC is required for RT should consider two dimensions: (i) *The legal dimension*. In interorganizational software projects, clients may claim the product was not delivered with the expected quality, and they may raise bugs even if the software is working in accordance to client's specifications.³² In order to avoid expensive disputes, the supplier should prove the correctness of the software and this can only be done by tracing the requirements specified by the clients with implementation/verification artifacts of suppliers.³² (ii) *BC as a platform*. An ecosystem for developing complex and large-scale software systems can be created among different parties that want to join and participate in any of the SDLC phases, such as customers, distributed teams composed of requirements engineers, developers, testers, third party vendors, crowd-workers, and regulators. Furthermore, our results indicate the need to make a set of decisions (see Section 4.1.2), for instance, how to link tools used throughout the SDLC with BC, and how to ensure that the delay of certification for transactions and data stored on the distributed ledger does not impact the quality and time-to-market of the software under development.

After identifying the need for BC in interorganizational software projects, the organization should determine the benefits along with the challenges associated with the implementation of this technology. Due to its inherent features, BC is expected to provide a holistic and reliable view of artifacts and trace links to all distributed stakeholders, incentivize software practitioners to create quality trace links, increase the use of traceability to support SDLC tasks and consequently, improve the performance of practitioners in addressing SDLC tasks. The increased quality and completeness of traceability may lead to increased software quality, consequently, less need for software maintenance and cost savings.¹²

Despite these benefits, organizations should be aware of the challenges associated with the implementation of this technology into their organizational environment. Our results revealed economic challenges in particular costs for the infrastructure (storage, network communication, and hardware) and for the development of applications for practitioners who participate in SDLC activities, and also for entities such as customers and regulators who need to access the data stored on the distributed ledger to verify requirements coverage, monitor project's progress and assess compliance to regulations. In addition, the results revealed that the implementation of BC in organizational settings is prone to organizational resistance to change, which is caused by three main factors: innovation-production gap, conservative management, and centralized mentality. These factors are related to the poor understanding of BC; therefore, education has been outlined as a critical success factor that can minimize the organizational resistance to change by changing the centralized mindset, improving the understanding of BC business values, and enhancing BC technical skills. To address the innovation-production gap, the findings suggest involving technicians working in production in the prototype testing process, which is led by the innovation team. In addition, the experts mentioned business, regulation, and technical concerns such as interoperability, security, and scalability issues. Although the category of business challenges emerged, the authors perceive this factor as not critical when applying BC for RT because they do not expect business transformation or new business models. As EX4 stated, "if you are trying to only achieve a kind of track and trace database, do not have high hopes of amazing transformative stuff" Furthermore, the experts encouraged organizations to continuously monitor the compliance of BC solutions with domain-specific regulations. A particular focus needs to be paid to the (mis)alignment between GDPR principles and inherent BC features.

The experts also recommended organizations to be agile, to experiment in an iterative fashion, and to set up dedicated teams working as a task force for the development and implementation of BC solutions. Given the globally distributed nature of organizations that may be interested in implementing BC technology, scaling agile is important.⁵⁷ Therefore, frameworks for scaling agile, such as Scaled Agile Framework (SAFe) can be adopted.⁵⁷ Principles of SAFe concepts, in particular, agile release trains (ART) are aligned with BC principle of breaking silos that exist within an organization or across organizational boundaries. In the case of BC-based SE solutions, ARTs should consist of dedicated teams with all the capabilities (e.g., SE, RT, and BC expertise) necessary to define, deliver and operate BC solutions. The experts also advocated creating a collaborative environment among different parties such as organizations interested to join the BC ecosystem, collaboration with BC startups, BC experts, and academic researchers who may guide the implementation process. However, due to the scarce market of BC experts, it is highly recommended to upskill existing personnel and acquire BC knowledge in-house.

5.2.3 | Traceability information model and incentive policies

This phase consists of defining the traceability information model (TIM), which has been advocated as a best practice by traceability researchers.^{58,59} The main utility of TIM lies in guiding the setup of traceability and enabling the validation of changes.⁵⁸ In addition, implementing traceability in an agreed manner ensures the consistency of results in multi-stakeholder projects.⁵⁸ A basic TIM defines types of artifacts to be traced and their metadata, trace links based on source artifact ID and destination artifact ID, and trace type rules to generate the semantics of trace links.⁵⁶ SC can be used to register only the artifacts and trace links that are defined in TIM and identify the semantics of trace links automatically. Further, organizations interested in implementing BC to enhance the motivation of practitioners to engage in traceability tasks need to define incentive policies that focus on the eligibility to create trace links, the validation of trace links quality and the amount of incentive that goes for the creation of quality trace links according to their priority.¹²

5.2.4 | BC platform selection

The experts provided guidance to the BC platform selection process that can be useful for organizations interested in implementing BC for RT. The results indicate network accessibility and programmability as critical elements that need to be considered. The selection between private and public BC platforms depends on the type of software under development. For instance, public platforms may be suitable in the case of open-source software with diverse and unknown contributors, while private platforms are more aligned with complex and large-scale software being developed by known organizations or distributed teams. The latter is the main focus of the BC proposal in Figure 4. Furthermore, programmability is another important factor to consider because the BC proposal makes use of SC to enable a set of functions. Therefore, it is necessary to choose a platform that supports SC execution.

The results suggested other factors to be considered, such as the alignment between foundational organizational values and BC platforms, transaction fees, consensus mechanism, and community of developers. Additionally, concerns were raised by the experts regarding the difficulty in selecting the best fitting BC platform due to the emergence of new platforms. A cost-effective practice advocated by the experts to facilitate the selection process is to continuously monitor what BC platforms other organizations or developers are choosing and “follow the stream.”

5.2.5 | Prototype

Our findings revealed that the innovation team of the interested organization is expected to test out a BC-enabled RT prototype. However, according to the experts, designing the right BC solution is not trivial. Figure 4 depicts a BC-enabled solution for RT, which entails four main components: (i) *Data collection*. The SDLC is composed of a variety of tools that generate artifacts that need to be traced, according to TIM. Data ingestion tools/plugins can be used to capture these artifacts. In addition, eligible stakeholders can create trace links and register these links on BC by means of SC. (ii) *Storage layer and smart contracts*. In this proposal, SC enable the following functions: register artifacts (id, type, name, description, priority, and parent_id), register trace links (source_artifact_id and dest_artifact_id), validate the quality of trace links, and reward the creators of quality trace links by means of digitized tokens based on incentive policies that are encoded into SC. (iii) *Query layer*. This layer enables traceability-related queries that comprise primitive links between adjacent artifact types in TIM and composite links between non-adjacent artifact types. (iv) *Presentation layer*. This layer enables the visualization of traceability information in an interactive and hierarchical manner to facilitate the comprehension of the overall system. For a more detailed description of the proposal, the authors refer to their previous study.¹²

Finally, based on the recommendations of our BC experts, organizations need to perform a performance analysis of the solution (latency, transaction throughput, and transaction speed), security analysis if the software is built for safety-critical systems, and user-interaction analysis to evaluate how stakeholders of the SDLC interact with BC.

5.2.6 | Production and scaling

In this phase, the prototype shifts into production. However, as confirmed by the experts, most of BC projects stop at the pilot phase. The interested organization should be aware that the main problem lies in the resistance of practitioners working in production because they have not been included in the pilot phase carried out by the innovation team or they have decided to not participate. Therefore, our results suggested close collaboration between the innovation and production teams during all the phases of the BC project. The inclusion of the production team in the early stages of the project may minimize their resistance and bridge the innovation-production gap, as referred to in this study.

5.3 | Limitations

The main limitation of this study is limited generalizability. This study does not claim that its results are generalizable due to the low number of interviews. The authors noticed that after eight interviews, no new concepts emerged; however, they decided to perform two more interviews to validate the theoretical saturation point. Although the theoretical saturation point was reached, it does not ensure generalizability. To address this, the authors selected a diverse set of participants in terms of job positions, projects domains and countries (see Table 2). Nonetheless, limited generalizability is a known limitation of qualitative research because qualitative research aims to enable a contextualized understanding of human experience.⁶⁰ While it is true that GT ensures the discovery of high-level concepts that are not specific to a subject or setting,⁶¹ it is also true that any qualitative inquiry method may be subject to external validity threats.⁶⁰ Furthermore, Myers⁶² claimed that novice researchers may find the coding process to be time-consuming and tiring. Consequently, they are prone to feeling lost in the coding process and unable to unveil concepts and categories grounded in data. As recommended by Annells⁶³ and Chun Tie et al.,³⁴ the first author of this study was assisted in her journey of inquiry by the other more experienced authors.

6 | CONCLUSION AND FUTURE WORK

This study adopted a GT approach to unveil the BC implementation process in organizational settings by performing semi-structured and in-depth interviews with BC experts. BC experts were selected through a rigorous selection process that put an emphasis on their diversity in terms of job positions, projects domains, and countries. The results revealed key activities, success factors, and core challenges, along with 17 concepts to explain the phenomenon under study.

The authors used the emerged concepts to improve a BC-enabled framework for RT in interorganizational software projects. The results suggested that the implementation process follows the following phases in an iterative manner: identify the need for distributed RT, perform a feasibility analysis, define the TIM and incentive policies, select the best fitting BC platform, test through prototypes, and shift from prototyping to production and scaling. These results may enhance the knowledge regarding this publicized technology and may encourage the collaboration between BC experts and requirements engineers towards developing more prototypes and proof-of-concepts.

Future efforts can be devoted to the following dimensions: (i) evolving the categories and concepts by collecting new data from a larger set of BC experts and other data collection techniques, such as observation; (ii) focusing on one of the emerged categories in-depth and using GT techniques to provide an explanatory theory, given that this study takes a general perspective on the implementation process of BC; and (iii) applying the results of this study in different domains and identifying similarities and discrepancies of the implementation process. Our main priority will be the development of the proposed BC-enabled RT prototype and the validation of the improved framework by requirements engineering experts.

ACKNOWLEDGEMENT

This work is part of a PhD project, funded by Østfold University College.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in figshare at <https://doi.org/10.6084/m9.figshare.19078079>.

ORCID

Selina Demi  <https://orcid.org/0000-0001-5988-4697>

Mary Sánchez-Gordón  <https://orcid.org/0000-0002-5102-1122>

REFERENCES

1. Duy PT, Hien DTT, Hien DH, Pham V-H. A survey on opportunities and challenges of Blockchain technology adoption for revolutionary innovation. In: *Proceedings of the Ninth International Symposium on Information and Communication Technology*. New York, NY, USA: Association for Computing Machinery; 2018:200-207.
2. Swan M. *Blockchain: Blueprint for A New Economy*. O'Reilly Media, Inc.; 2015.
3. Frizzo-Barker J, Chow-White PA, Adams PR, Mentanko J, Ha D, Green S. Blockchain as a disruptive technology for business: a systematic review. *Int J Inform Manag*. 2020;51:102029. doi:10.1016/j.ijinfomgt.2019.10.014
4. Zheng Z, Xie S, Dai H-N, Chen X, Wang H. Blockchain challenges and opportunities: a survey. *Int J Web Grid Services*. 2018;14(4):352-375. doi:10.1504/IJWGS.2018.095647
5. Belotti M, Božić N, Pujolle G, Secci S. A Vademecum on blockchain technologies: when, which, and how. *IEEE Commun Surveys Tutorials*. 2019;21(4):3796-3838. doi:10.1109/COMST.2019.2928178
6. Flovik S, Moudnib RA, Vassilakopoulou P. Determinants of blockchain technology introduction in organizations: an empirical study among experienced practitioners. *Proc Comput Sci*. 2021;181:664-670. doi:10.1016/j.procs.2021.01.216

7. Janssen M, Weerakkody V, Ismagilova E, Sivarajah U, Irani Z. A framework for analysing blockchain technology adoption: integrating institutional, market and technical factors. *Int J Inform Manag.* 2020;50:302-309. doi:10.1016/j.ijinfomgt.2019.08.012
8. 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.
9. Dozier PD, Montgomery TA. Banking on blockchain: an evaluation of innovation decision making. *IEEE Trans Eng Manag.* 2020;67(4):1129-1141. doi:10.1109/TEM.2019.2948142
10. Hughes L, Dwivedi YK, Misra SK, Rana NP, Raghavan V, Akella V. Blockchain research, practice and policy: applications, benefits, limitations, emerging research themes and research agenda. *Int J Inform Manag.* 2019;49:114-129. doi:10.1016/j.ijinfomgt.2019.02.005
11. Demi S, Colomo-Palacios R, Sánchez-Gordón M. Software engineering applications enabled by blockchain technology: a systematic mapping study. *Appl Sci.* 2021;11(7):2960. doi:10.3390/app11072960
12. Demi S, Sánchez-Gordón M, Colomo-Palacios R. A blockchain-enabled framework for requirements traceability. In: Yilmaz M, Clarke P, Messnarz R, Reiner M, eds. *Systems, Software and Services Process Improvement*. Cham: Springer International Publishing; 2021:3-13. doi:10.1007/978-3-030-85521-5_1
13. Colomo-Palacios R. Cross fertilization in software engineering. In: Yilmaz M, Niemann J, Clarke P, Messnarz R, eds. *Systems, Software and Services Process Improvement*. Cham: Springer International Publishing; 2020:3-13. doi:10.1007/978-3-030-56441-4_1
14. 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)*; 2018:1-1.
15. Lenarduzzi V, Lunesu MI, Marchesi M, Tonelli R. Blockchain applications for agile methodologies. In: *Proceedings of the 19th International Conference on Agile Software Development: Companion*. Porto, Portugal: Association for Computing Machinery; 2018:1-3.
16. Beller M, Hejderup J. Blockchain-based software engineering. In: *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results*. Montreal, Quebec, Canada: IEEE Press; 2019:53-56.
17. Yilmaz M, Tasel S, Tuzun E, Gulec U, O'Connor RV, Clarke PM. Applying blockchain to improve the integrity of the software development process. In: Walker A, O'Connor RV, Messnarz R, eds. *Systems, Software and Services Process Improvement*. Cham: Springer International Publishing; 2019:260-271. doi:10.1007/978-3-030-28005-5_20
18. Bose RPJC, Phokela KK, Kaulgud V, Podder S. BLINKER: a blockchain-enabled framework for software provenance. In: *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*; 2019:1-8.
19. Yau SS, Patel JS. Application of blockchain for trusted coordination in collaborative software development. In: *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*; 2020:1036-1040.
20. Singi K, Kaulgud V, Chandra Bose RPJ, 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)*; 2020:37-47.
21. Pohl K. *Requirements Engineering: Fundamentals, Principles, and Techniques*. 1st ed. Springer Publishing Company, Incorporated; 2010. doi:10.1007/978-3-642-12578-2
22. Winkler S, von Pilgrim J. A survey of traceability in requirements engineering and model-driven development. *Softw Syst Model.* 2010;9(4):529-565. doi:10.1007/s10270-009-0145-0
23. Demi S, Sanchez-Gordon M, Colomo-Palacios R. What have we learnt from the challenges of (semi-) automated requirements traceability? A discussion on blockchain applicability. *IET Softw.* 2021;15(6):391-411.
24. Gotel OCZ, Finkelstein CW. An analysis of the requirements traceability problem. In: *Proceedings of IEEE International Conference on Requirements Engineering*; 1994:94-101.
25. Gotel O, Cleland-Huang J, Hayes JH, et al. Traceability fundamentals. In: Cleland-Huang J, Gotel O, Zisman A, eds. *Software and Systems Traceability*. London: Springer; 2012:3-22. doi:10.1007/978-1-4471-2239-5_1
26. Mäder P, Egyed A. Do developers benefit from requirements traceability when evolving and maintaining a software system? *Empir Software Eng.* 2015;20(2):413-441. doi:10.1007/s10664-014-9314-z
27. Regan G, McCaffery F, McDaid K, Flood D. The barriers to traceability and their potential solutions: towards a reference framework. In: *2012 38th Euromeric Conference on Software Engineering and Advanced Applications*; 2012:319-322.
28. Murugappan S, Prabha D. Requirement traceability for software development lifecycle. *Int J Sci Eng Res.* 2017;8:1-11.
29. Maro S, Steghöfer J-P, Staron M. Software traceability in the automotive domain: challenges and solutions. *J Syst Softw.* 2018;141:85-110. doi:10.1016/j.jss.2018.03.060
30. Wohrlab 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. *Requirements Eng.* 2018;25(1):21-45. doi:10.1007/s00766-018-0306-1
31. Jhala KS, Oak R, Khare M. Smart collaboration mechanism using Blockchain technology. In: *2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*; 2018:117-121.
32. Rempel P, Mäder P, Kuschke T, Philippow I. Requirements traceability across organizational boundaries—a survey and taxonomy. In: Doerr J, Opdahl AL, eds. *Requirements Engineering: Foundation for Software Quality*. Berlin, Heidelberg: Springer; 2013:125-140. doi:10.1007/978-3-642-37422-7_10
33. Glaser BG, Strauss AL. *The discovery of grounded theory: strategies for qualitative research*. Vol. 17. Aldine Publishing Company; 1967:364. doi:10.1097/00006199-196807000-00014
34. Chun Tie Y, Birks M, Francis K. Grounded theory research: a design framework for novice researchers. *SAGE Open Med.* 2019;7:1-8. doi:10.1177/2050312118822927
35. Urquhart C, Fernandez W. Grounded theory method: the researcher as blank slate and other myths. In: *ICIS 2006 Proceedings*; 2006.
36. Wiesche M, Jurisch MC, Yetton PW, Krmar H. Grounded theory methodology in information systems research. *MIS Q.* 2017;41(3):685-701. doi:10.25300/MISQ/2017/41.3.02
37. Coleman G, O'Connor R. Using grounded theory to understand software process improvement: a study of Irish software product companies. *Inform Softw Technol.* 2007;49(6):654-667. doi:10.1016/j.infsof.2007.02.011
38. Hoda R, Noble J, Marshall S. Using grounded theory to study the human aspects of software engineering. In: *Human Aspects of Software Engineering*. New York, NY, USA: Association for Computing Machinery; 2010:1-2.
39. Stol K-J, Ralph P, Fitzgerald B. Grounded theory in software engineering research: a critical review and guidelines. In: *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*; 2016:120-131.

40. Glaser BG. *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory*. Sociology Press; 1978.
41. Li M, Smidts CS. A ranking of software engineering measures based on expert opinion. *IEEE Trans Softw Eng*. 2003;29(9):811-824. doi:10.1109/TSE.2003.1232286
42. Fehring RJ. *Methods to Validate Nursing Diagnoses*. Nursing Faculty Research and Publications; 1987:27.
43. Herranz E, Palacios RC, de Amescua SA, Sánchez-Gordón M-L. Towards a gamification framework for software process improvement initiatives: construction and validation. *J UCS*. 2016;22:1509-1532.
44. Miles MB, Huberman AM. *Qualitative Data Analysis: An Expanded Sourcebook*. SAGE; 1994.
45. Baltés S, Ralph P. Sampling in software engineering research: a critical review and guidelines. *Empir Softw Eng*. 2021;27(4):1-31. doi:10.1007/s10664-021-10072-8
46. Demi S, Sánchez-Gordón M, Kristiansen M. Blockchain for requirements traceability: a qualitative approach. https://figshare.com/articles/dataset/Blockchain_for_Requirements_Traceability_A_Grounded_Theory_Approach/19078079. Accessed 10 Jun 2022
47. Oates BJ. *Researching Information Systems and Computing*. SAGE; 2006.
48. Recker J. *Scientific Research in Information Systems: A Beginner's Guide*. Berlin Heidelberg: Springer-Verlag; 2013. doi:10.1007/978-3-642-30048-6
49. Javdani Gandomani T, Ziaei Nafchi M. An empirically-developed framework for agile transition and adoption: a grounded theory approach. *J Syst Softw*. 2015;107:204-219. doi:10.1016/j.jss.2015.06.006
50. Farshidi S, Jansen S, España S, Verkleij J. Decision support for blockchain platform selection: three industry case studies. *IEEE Trans Eng Manag*. 2020; 67(4):1109-1128. doi:10.1109/TEM.2019.2956897
51. Zhou Y, Soh YS, Loh HS, Yuen KF. The key challenges and critical success factors of blockchain implementation: policy implications for Singapore's maritime industry. *Mar Policy*. 2020;122:104265. doi:10.1016/j.marpol.2020.104265
52. Walsh C, O'Reilly P, Gleasure R, et al. Understanding manager resistance to blockchain systems. *Eur Manag J*. 2021;39(3):353-365. doi:10.1016/j.emj.2020.10.001
53. Hastig GM, Sodhi MS. Blockchain for supply chain traceability: business requirements and critical success factors. *Prod Oper Manag*. 2020;29(4): 935-954. doi:10.1111/poms.13147
54. Biswas B, Gupta R. Analysis of barriers to implement blockchain in industry and service sectors. *Comput Ind Eng*. 2019;136:225-241. doi:10.1016/j.cie.2019.07.005
55. Belchior R, Vasconcelos A, Guerreiro S, Correia M. A survey on blockchain interoperability: past, present, and future trends. *ACM Comput Surv (CSUR)*. 2021;54(8):1-41. doi:10.1145/3471140
56. 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)*; 2018:269-276.
57. Paasivaara M. Adopting SAFe to scale agile in a globally distributed organization. In: *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*; 2017:36-40.
58. 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*; 2009:21-25.
59. Cleland-Huang J, Berenbach B, Clark S, Settimi R, Romanova E. Best practices for automated traceability. *Computer*. 2007;40(6):27-35. doi:10.1109/MC.2007.195
60. El Hussein M, Hirst S, Salyers V, Osuji J. Using grounded theory as a method of inquiry: advantages and disadvantages. *Qual Rep*. 2014;19:1-15. doi:10.46743/2160-3715/2014.1209
61. Glaser BG. *Doing grounded theory: issues & discussion*. Mill Valley, Calif: Sociology Pr; 1998.
62. Myers MD. *Qualitative Research in business and management*. SAGE; 2019.
63. Annells M. Grounded theory method: philosophical perspectives, paradigm of inquiry, and postmodernism. *Qual Health Res*. 1996;6(3):379-393. doi:10.1177/104973239600600306

How to cite this article: Demi S, Sánchez-Gordón M, Kristiansen M. Blockchain for requirements traceability: A qualitative approach. *J Softw Evol Proc*. 2022;e2493. doi:10.1002/smr.2493






PAPER VI

S. Demi, R. Colomo-Palacios, M. Sánchez-Gordón, C. Velasco, and R. Cano, "*A Neural Blockchain for Requirements Traceability: BC4RT Prototype*," in Systems, Software and Services Process Improvement: 29th European Conference, EuroSPI 2022, Salzburg, Austria, August 31–September 2, 2022, Proceedings, pp. 45–59, Springer, 2022.



A Neural Blockchain for Requirements Traceability: BC4RT Prototype

Selina Demi¹ , Ricardo Colomo-Palacios¹ , Mary Sánchez-Gordón¹ ,
Carlos Velasco², and Ramon Cano²

¹ Østfold University College, Halden, Norway
{selina.demi, ricardo.colomo-palacios,
mary.sanchez-gordon}@hiof.no

² ByEvolution Creative Factory, Málaga, Spain
{carlos.velasco, ramon.cano}@byevolution.com

Abstract. The ever-increasing globalization of the software industry presents challenges related to requirements engineering activities. Managing requirements' changes and tracing software artifacts is not trivial in a multi-site environment composed of a variety of stakeholders that do not trust each other. In this study, we propose a neural blockchain prototype for the traceability of requirements (BC4RT) throughout the software development lifecycle in interorganizational software projects. The prototype is implemented using a neural blockchain platform, namely NDL ArcaNet, due to its inherent properties: performance efficiency, sustainability, and scalability. Besides these features, the proposed prototype provides a holistic and reliable view of software artifacts, requirements' changes, and trace links. The increased visibility enhances collaboration, communication, and trust among stakeholders, and can potentially improve software development efficiency and quality.

Keywords: Blockchain technology · Requirements traceability · Interorganizational software projects · Neural distributed ledger

1 Introduction

Software engineering (SE) has shifted from conventional co-located development to global distributed development. Today's software products are developed as a result of complex supply chains that entail the collaboration of a variety of distributed partners throughout the software lifecycle, from conceptualization and development, to maintenance [1]. While global software development companies leverage benefits of distributed development: time, cost, and access to skillful resources, they also face a set of challenges: lack of communication and coordination, lack of uniform processes in a multi-site environment, lack of trust, lack of management and transfer, and challenges related to requirements engineering (RE) activities [2] which is the focus of this study. Managing requirements' changes, and tracing software artifacts in both a forward and backward direction is not a trivial activity in interorganizational software projects [3]. Although

a plethora of traceability studies exists [4], the traceability community has outlined the open challenge of enabling full traceability in complex and large-scale software development contexts that rely on cross-organizational collaboration of multiple stakeholders [5, 6].

This study proposes a neural blockchain prototype for the trustworthy management and traceability of requirements in interorganizational software projects. This proposal lies in the concept of creating tokens for each requirement, tracking the lifecycle of such tokens, and certifying operations that are performed on tokens, without the need for resource-wasteful consensus algorithms. Therefore, neural blockchains present an opportunity to store artifacts created throughout the software development lifecycle in a scalable, efficient, and transparent manner, while retaining security. In addition, the proposed prototype enables participants of the software development lifecycle with a holistic and reliable view of software artifacts, requirements' changes, and trace links. The increased visibility on the software development process may lead to enhanced communication and coordination, and trust among stakeholders in interorganizational software projects. In turn, this can potentially improve software development efficiency and quality.

The remainder of this study is structured as follows: Sect. 2 provides an overview of the fundamental blockchain concepts, applications of blockchain technology in software engineering, and requirements engineering and traceability challenges. Section 3 proposes a neural blockchain prototype for the management and traceability of requirements throughout the software development lifecycle, and Sect. 4 presents implementation details of the prototype. Section 5 concludes the study and presents directions for future research.

2 Background

2.1 Blockchain Basics

Blockchain is a peer-to-peer (P2P) distributed ledger technology that stores digital transactions in a chain of blocks [7]. These digital transactions represent interactions between P2P network peers that entail the exchange of digital assets which can be in the form of information, good, services or rules to trigger another transaction [8]. Network peers group up the transactions into blocks and distribute them throughout the network. It is noteworthy that these peers need to achieve agreement with regards to the correct data state on the system. Ensuring the consistency of data on the ledger for all network peers requires the deployment of consensus algorithms which vary among different blockchain implementations. The main two groups of consensus algorithms are [8]: (i) Proof-of-X algorithms, and (ii) Byzantine Fault Tolerant algorithms. Furthermore, the exchange of assets relies on contractual rights and obligations of nodes that can be digitized and managed by smart contracts (SCs). SCs are computer programs that are stored on the blockchain and enable the modification of the ledger state when certain conditions are met. The modification of the ledger state is triggered by a transaction posted to the distributed ledger [9]. Initially, smart contracts were conceptualized to enable trusted agreements among different parties in a trustless environment [9], but nowadays they

are considered similar to general purpose software programs and can, at least theoretically, perform any computational task that can be performed by conventional programs [10].

The first blockchain application was proposed in 2008 and was named Bitcoin [11]. Although distributed ledger technologies existed prior to Bitcoin, the novelty of blockchain lies in the combination of existing technologies, such as P2P networks, cryptography, transactions timestamping and shared computational power [8]. The combination of these technologies enables the sharing and storage of data in a decentralized manner without the need to entrust a central party for the maintenance of the ledger. Belotti et al. [8] categorized blockchains with respect to network accessibility in: (i) permissionless blockchains – anyone can participate in the network and modify the network state, e.g., Bitcoin and Ethereum. (ii) permissioned blockchains – only selected nodes can participate in the network and modify the network state. The latter can be further categorized according to the nature of participants in private blockchains and consortium blockchains. While in private blockchains participants are within the same organization, in consortium blockchains several organizations share a common goal.

2.2 Blockchain in Software Engineering

Recently, academic researchers have encouraged the cross-fertilization of blockchain technology and SE [12, 13]. Our previous systematic mapping study [14] explored the alignment between blockchain inherent properties and the modern (global) SE landscape, benefits and challenges of using this technology, and the proposed use cases. In what follows, a limited number of these use cases is introduced:

Lenarduzzi et al. [15] proposed a blockchain model that uses SCs to relieve some of the duties of the product owner in agile processes such as Lean-Kanban or Scrum. In this model, SCs automatically validate the correctness of user stories implemented by developers by comparing the acceptance tests output with the expected output. The correct implementation of user stories triggers the automatic payment to developers in cryptocurrencies or tokens.

Yilmaz et al. [16] proposed a blockchain model in which the project leader introduces new work structures to the blockchain network, developers choose their preferred tasks and develop code which is validated by testers. Testers share a candidate block and generate consequent blocks collaboratively. This model is aimed to address trust and integrity issues in large-scale agile development.

Bose et al. [17] proposed the application of blockchain for trustworthy software provenance. The authors introduced a framework enabled by blockchain technology named Blinker that captures and queries provenance data by means of PROV family of specifications, verifies the authenticity of the data through voting mechanisms and enables hierarchical and interactive visualization of provenance related data.

Yau and Patel [18] adopted blockchain technology to achieve reliable coordination in collaborative software development. Their blockchain-based approach aims to address limitations of centralized solutions, such as single point of failure, data tampering and auditability, and lack of verification for the data to be stored. Smart contracts are used to verify the compliance of acceptance criteria for software components in an automatic fashion.

None of these studies focus on the application of blockchain technology for the management and traceability of requirements throughout the software development lifecycle in interorganizational software projects.

2.3 Requirements Engineering and Traceability

Requirements engineering (RE) is a critical component of effective software development projects. While previous studies provided empirical evidence to support the contribution of effective RE to improved productivity, product quality, and risk management [19], the RE process has been considered as inherently complex and difficult to standardize via holistic solutions [20]. As software becomes more complex and the number of stakeholders, along with their heterogeneity increases, there is a need to enhance the large-scale RE process [21]. One of the most critical challenges that has been identified in RE, particularly in managing requirements' changes in global software development is the lack of communication, coordination, and control that leads to reduced levels of trust and confidence among distributed team members [22]. In addition, Akbar et al. [22] highlighted the lack of change impact analysis at distributed sites as a significant challenge. Estimating the impact of changes on the system's costs, time and quality is essential, yet difficult to achieve in distributed settings.

According to Jayatilleke and Lai [23], requirements traceability can contribute to keep track of the impact of changes. Traceability has been defined as “*the ability to follow the life of a requirement in both a forward and backward direction...*” [24] or as the ability to create, maintain, and use links between artifacts generated in different phases of the software lifecycle [5]. Traceability is particularly important in safety-critical domains, in light of proving the specification of safety requirements, the consideration of these requirements during the design and development phases, and their validation in test cases [25]. Despite its importance, establishing traceability in practice is not a trivial task [25]. Our recent systematic literature review reported on 21 challenges of implementing traceability in organizational settings [4]. In particular, the findings revealed that in practice, traceability is perceived as an overhead, and its potential benefits are invisible throughout the software development lifecycle. Previous studies [6, 25] pinpointed the provider-user gap as the main factor that shapes this perception, along with the poor visualization of trace links. As a result, practitioners become demotivated to create and maintain trace links and assign a low priority to traceability tasks. In addition, previous studies [6, 25, 26] raised concerns regarding the deterioration of trace links as a consequence of not updating these links when artifacts change. These changes should be propagated and affected stakeholders should be notified in order to update the corresponding trace links.

The global software development paradigm exacerbates these issues, as the communication, coordination, and trust among stakeholders is difficult to achieve in distributed settings [4]. One of the few studies that provides empirical evidence on requirements traceability in interorganizational software projects has been carried out by Rempel et al. [3]. Rempel et al. [3] outlined organizational boundaries as the main problem area, as it leads to restricted access to artifacts created by the other project parties due to lack of trust. Therefore, the authors outlined the need to ensure availability and reliability of traceability in interorganizational software projects. To address these requirements, our

study proposes a blockchain-enabled prototype for requirements traceability (BC4RT) which is described in Sect. 3.

3 Blockchain-Enabled Requirements Traceability Prototype

Managing and tracing requirements throughout the software development lifecycle in a transparent and reliable fashion is important to ensure trust among different stakeholders. Figure 1 depicts a simplified version of the software development lifecycle which consists of 4 logical users – requirements manager, developer, tester, and customer. Other users are omitted for simplicity.

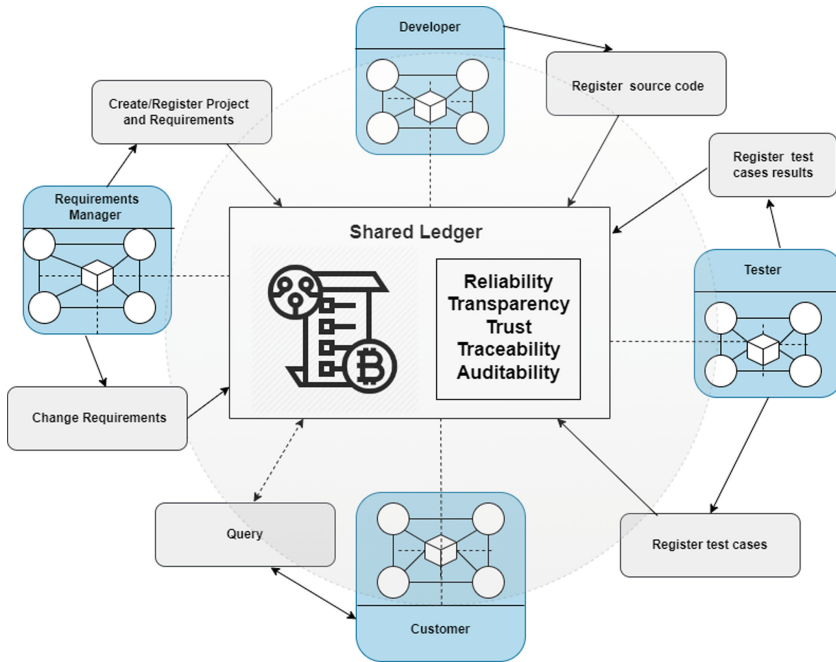


Fig. 1. High-level conceptualization of blockchain-enabled requirements traceability prototype: BC4RT

This prototype relies on the assumptions that users are located in distributed settings, and they do not trust each other, but they need to collaborate for the development of a large-scale software development project. In this context, blockchain technology can serve as a secure repository to store software artifacts and their changes by ensuring reliability, transparency, trust, traceability, and auditability. The logical users can perform different operations which are explained in the following section.

Requirements managers can create or register new projects and new requirements for each project that should be stored on the distributed ledger. The timestamp of when the requirement was created, contributor name, and the current status “created” should also be stored on the ledger. In addition, requirements managers should be able to change

existing requirements and their respective attributes, such as version, description, short name. In such a case, the current status of the requirement should be “*changed*” from “*created*” and the timestamp of when the requirement is changed should be stored on the ledger. However, the immutable nature of blockchain technologies does not allow changing stored data.

At first glance, one may argue that the immutable property of blockchain goes against the ever-changing nature of software artifacts. The authors identified two potential solutions to address this challenge: (i) when requirements managers change existing requirements, a new requirement record with a new ID is created. This new requirement should point to the initial requirement that was changed by means of a previous requirement ID field; (ii) perceive requirements as digital assets and using the concept of tokens to represent them. Each token may generate its own blockchain ledger to audit the lifecycle of any requirement token throughout the software development lifecycle. In this study, the authors followed the latter approach, as it is more efficient than the former.

Furthermore, developers can register source code files for each specific requirement and consequently, the current status of the requirement is updated from “*created*” or “*changed*” to “*implemented*”. Testers can register test cases for each requirement and the results of these test cases. The registration of test cases changes the status of the requirement automatically from “*implemented*” to “*tested*”. Moreover, the customer has permission to view requirements’ changes, and track requirements’ lifecycle using the audit mode. In addition, the customer can perform more complex queries, for instance retrieve the IDs and number of requirements whose status is “*tested*”, but the test result is “*failed*”.

Finally, it is important to consider an efficient, scalable and secure platform to store software artifacts, such as source code files, or test cases files. If conventional blockchain platforms were chosen, these files would have been stored in secure off-chain storage, such as IPFS (Interplanetary File System) and the generated hash would have been stored in the blockchain platform to access the file’s content [17]. This study adopts a novel blockchain platform that enables the secure storage of files of any size and type, while retaining efficiency and scalability. The blockchain platform adopted by this study is explained in the following Sect. 4.1.

4 Implementation

4.1 Neural Distributed Ledger

The concept of neural distributed ledger (NDL) was recently proposed by Velasco et al. [27] and inspired by Swan [28]’s idea of developing blockchains as “*personal thinking chains*”. A neural blockchain is internally structured into subsets of groups that work in parallel and are interconnected analogously to how neuron groups are aggregated in human brains. The main utility of such blockchains lies in addressing interoperability, performance, and scalability issues that exist in conventional blockchain platforms [27]. In this study, the authors decided to implement an innovative and collaborative P2P network, namely NDL ArcaNet. NDL ArcaNet ensures the protection and secure transfer of digital assets of any type.

In order to understand NDL ArcaNet, it is important to explain the concept of NDL Arca, as a secure token directory. NDL Arca [29] is a distributed repository of tokens that ensures the protection of tokens' content against illegitimate access. Tokens are valuable, unique and certified data that must be accessed only by their legitimate owners and must be stored throughout their lifecycle in a secure repository to prevent unauthorized access and illegitimate modifications. Tokens are grouped into tables which are in turn grouped into databases. This structure lies in the combination of key-value storage and column-based databases. The keys are valuable to enable the identification of contents in any environment and are expressed in the ULID (Universally Unique Lexicographically Sortable Identifier) format. ULID generates identifiers by considering both base32 encoded timestamp (first 10 characters), and randomness (remaining 16 characters). The values are always encrypted and point to dynamic tables (variable array []). The non-static columns of these tables contain token fields' ID and token fields' content. This dynamic nature enables token fields' values to be changed according to users' needs. CRUD (create, read, update, delete) operations can be performed on tokens, along with other operations, such as import and export.

Moreover, according to [29], the security of NDL Arca is ensured by applying a set of techniques, such as 2-key encrypted token, as the data is double encrypted with database password and token password, AES 256 (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman), zero trust and zero knowledge cryptography, and hashing functions. It is worthy to mention that although NDL Arca was designed mainly for a blockchain network due to its inherent capabilities of replication, hashing of contents, and distributing them across the network, it can be installed on any system according to [29], e.g., using Arca to create a centralized dedicated server, or a database system in the cloud. The use of NDL Arca in a multi-domain network is referred to as NDL ArcaNet.

In our case study, requirements are considered tokens because they are valuable, identifiable, and unique digital assets. Requirements tokens are stored in a secure token repository and are created and updated in a collaborative manner among different stakeholders of the software development lifecycle who share a secret key. Each operation applied on tokens is visible and transparent to other parties in the network. All the operations performed on tokens will be validated by trusted certifiers who are incentivized by means of service payments that they receive for each digital signature. Trusted certifiers will validate operations on tokens without knowing the content of tokens, by applying zero-knowledge cryptography.

The authors selected this platform due to three main advantages that are important in the software engineering context: (i) *performance efficiency* – each node (wallet) applies and verifies its own transactions independently, enabling parallel work, thus maximizing the number of transactions per second. Each node can trust the token content by checking signatures, removing the need for the majority of the network nodes to vote and reach a consensus. The lack of consensus leads to each wallet working as a local database, but with slightly higher latency due to the use of signature mechanisms. Should consensus-based distributed ledger technologies be used, storing a large number of requirements or other large software files would not be affordable. However, NDL systems scale better and their limitations regarding real time operations are comparable to the limitations of centralized databases. (ii) *sustainability* – nodes collaborate to validate transactions,

therefore costly, resource-wasteful, and competitive-based consensus algorithms (e.g., PoW, PoS) are not used and gas is not required to perform transactions. (iii) *scalability* – the platform can integrate million nodes because each node is independent and can work in real-time. While the Internet transfers packets of data, NDL ArcaNet transfers signed packets of data. Despite this, ArcaNet is able to scale in a similar fashion to the Internet.

4.2 Blocks Structures for BC4RT Prototype

The proposed blockchain-enabled requirements traceability prototype relies on the underlying blocks structures that are depicted in Fig. 2.

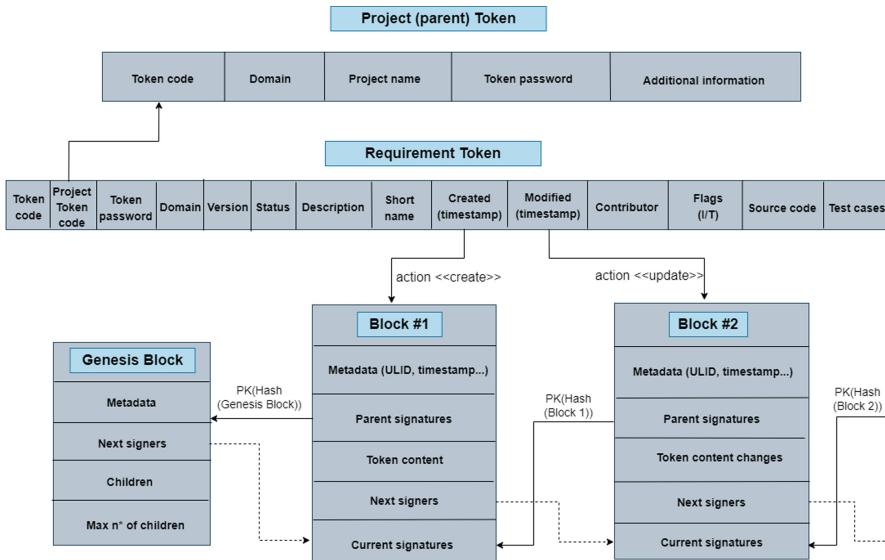


Fig. 2. Blocks structures for BC4RT prototype

Each token generates its own signed blockchain ledger that enables the verification of its provenance, integrity, evolution, and history, by means of the audit mode. The goal of the BC4RT prototype is to trace the lifecycle of requirements throughout the software development lifecycle. Therefore, a token was created for each requirement, as a child of the project token. The project token consists of the following fields: token code (ULID), domain, name of the project, and the password of the token. Other fields can be created to include additional information regarding the project. Furthermore, the requirement token consists of the following fields: token code (ULID), project code that points to the parent token, token password, domain, version of the requirement, the current status of the requirement (created, modified/changed, implemented, tested), requirement’s description, short name, timestamp of when the requirement was created, timestamp of when the requirement was modified, the contributor who performed a specific operation on the token, flags (implemented/tested), source code file, and test cases file.

The emission of the requirement token generates the first block (Block #1) which is composed of the following elements: metadata, e.g., ULID, and timestamp, previous block/parent signatures which entail signing with private keys the hash of the previous block, token content which consists of the fields' content of the requirement token, next signers or the signers of the next block which are a set of trusted certifiers and random nodes, and the signatures of the current block. Signers that are defined in the previous block's next signers field should sign with their private keys the hash of the current block fields (metadata, parent signatures, token content, and next signers). While the consequent blocks have the same structure as the first block, they do not store the whole content of token fields, only the changes.

Finally, it is noteworthy that any first block needs a genesis block which is provided by the other parties of the network in a random manner. This structure allows stakeholders of the software development lifecycle to keep track of what/when/how/by whom requirements were created, changed, implemented, and tested in a trustworthy and transparent manner. A shared traceability repository based on blockchain ensures that software artifacts stored by distributed stakeholders have not been altered illegitimately.

4.3 User Interface

In what follows, we present the front end of the BC4RT application using simple scenarios that rely on the iTrust application that can be accessed online [30]. iTrust is an electronic health records application that is developed and maintained as a software engineering project for undergraduate students at North Carolina State University [31]. iTrust was chosen because it deals with safety-critical information and due to the availability of the traceability dataset [31].

The user logs in by specifying the role, i.e., requirements manager, coder, tester, or customer, as depicted in Fig. 3. Figure 4, Fig. 5, and Fig. 6 show the view of the requirements manager who is allowed to create a new project, new requirement tokens, and update existing requirements, respectively. The attributes of requirements are requirement ID, contributor, requirement version, description, short name, current state, history of states (created, changed, implemented, tested), source code file, and test case file. Each of these tokens generates its own blockchain ledger that stores the changes that have been validated by trusted certifiers.

Fig. 3. Login view

First, the requirements manager creates a new project with the assigned project token ID: 01G4JM6G1FQPPHKAH4EAXNA27M (See Fig. 4). Then, the requirements manager creates new requirements for the project by clicking on the “Create” option of the radio button “Requirement”, inputs the description and short name of the requirement, while the token ID is generated automatically (See Fig. 5).

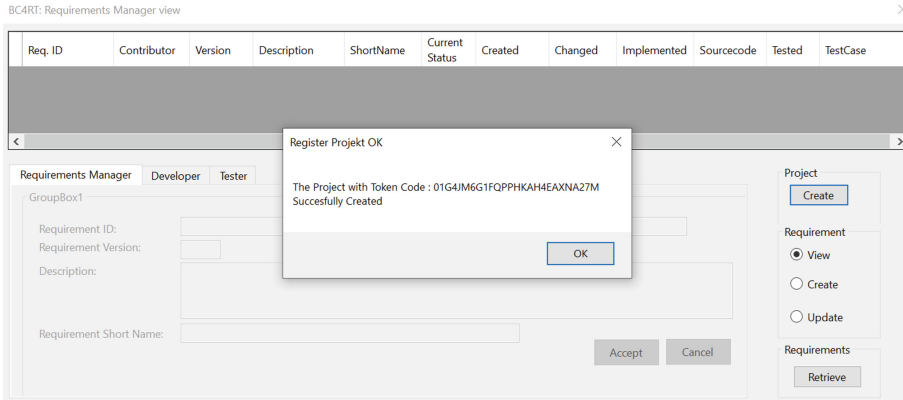


Fig. 4. Requirements manager view (“Create project”)

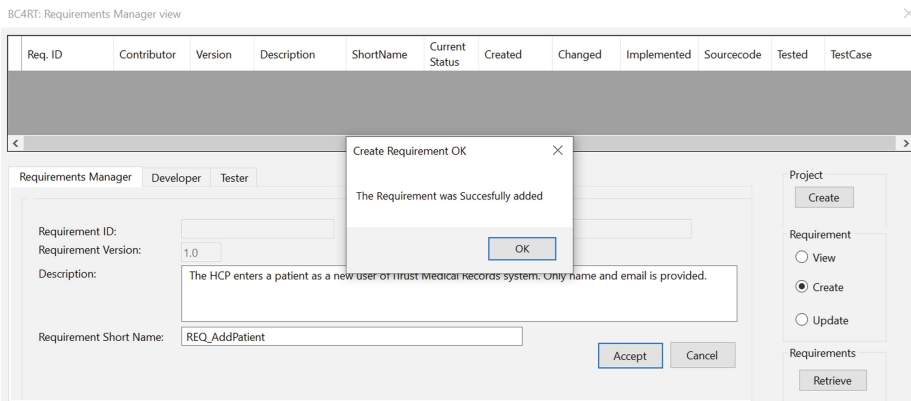


Fig. 5. Requirements manager view (“Create requirement”)

Once the requirements manager clicks “Accept”, the blockchain ledger is generated for the requirement token. The current state of the requirement is “created”, and the timestamp of when the requirement was created is also presented to the user (See Fig. 6). The requirements manager can also update previously-created requirements by clicking on the “Update” option of the radio button “Requirement”. For instance, in Fig. 6 the requirements manager is updating the requirement with the ID = 01G4JMRW7M6CZPXJK030H4AKK, by entering the new version = 1.1, description = “The patient should be able to view and edit lab procedure tasks” and short name =

“REQ_ViewEditLab”. Once the requirement manager clicks “Accept”, the requirement token fields are updated, the current status is “changed”, and the timestamp of when the requirement was changed is also stored and presented to the user, as depicted in Fig. 6.

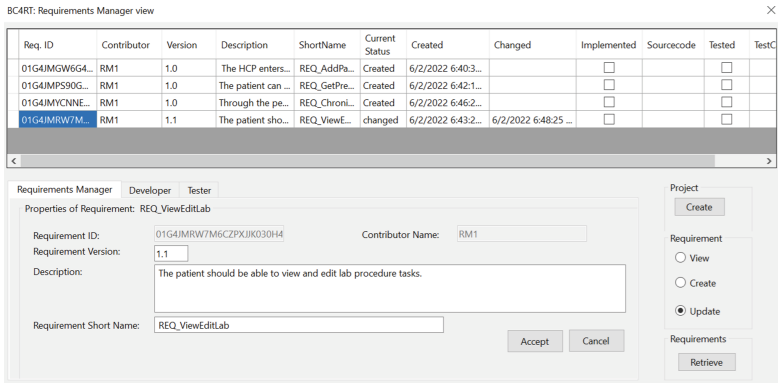


Fig. 6. Requirements manager view (“Update requirement”)

Second, the developer logs in the blockchain platform and is allowed to upload the source code file for each requirement (See Fig. 7). Once the developer enters a source code file and clicks on the “Accept” button, the state of the specific requirement token is updated in three dimensions: (i) *source code* field is updated with the name of the file or (ii) the *implemented* field is updated (iii) the *current status* is changed from “created” or “changed” into “implemented”.

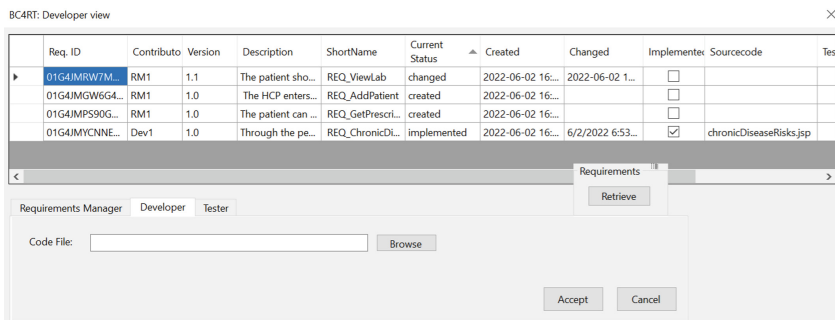


Fig. 7. Developer view (“Upload source code”)

Third, the tester logs in the blockchain platform and is allowed to upload the test case file for each requirement (See Fig. 8). Once the tester enters a test case file and clicks on the “Accept” button, the state of the specific requirement is updated in three dimensions: (i) *test case* field is updated with the test case file name (ii) the *current status* of the requirement is changed from “implemented” into “tested” (iii) the *tested* field is updated, if the tester clicks on the “Passed” option of the radio button “Test Result”.

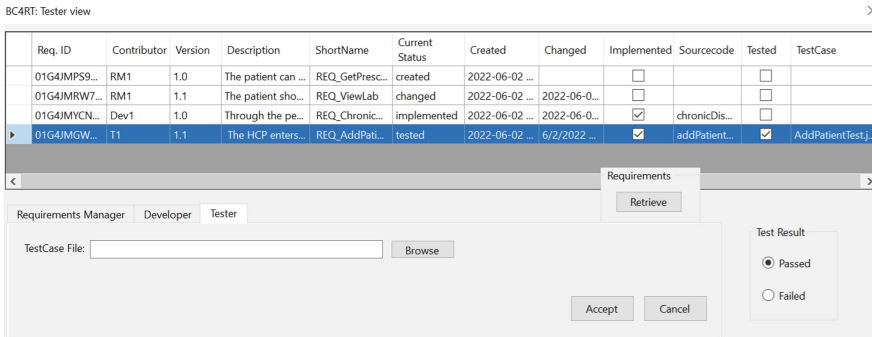


Fig. 8. Tester view (“Upload test cases and test results”)

Finally, the customer is constrained to only view the state of the project token and requirements tokens. Therefore, the customer can check the list of all requirements, which requirements have been created, changed, implemented, and/or tested. In addition, it is possible to trace the lifecycle of each requirement by double clicking on a specific requirement record. An example of the history of a specific requirement is depicted in Fig. 9.

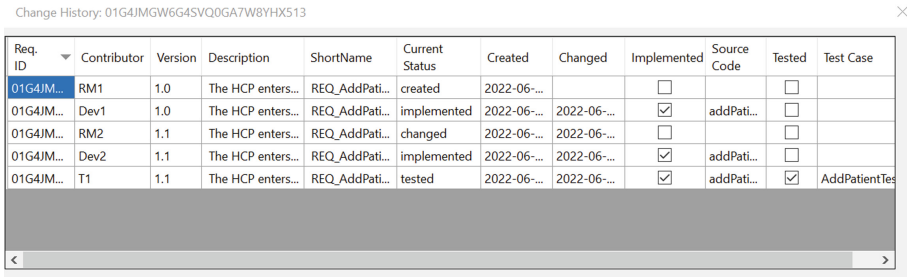


Fig. 9. Trace the lifecycle of a specific requirement

5 Conclusion and Future Work

This study presented a blockchain-oriented prototype, namely BC4RT to enable the traceability of software artifacts created by distributed stakeholders throughout the software development lifecycle. For each requirement stored on the distributed ledger, one could trace its origin, updates, the timestamp of when it was created or changed, if it was implemented and/or tested, and by whom, the current status, and related software artifacts, such as source code and test cases, in a scalable, efficient and trustworthy manner. Therefore, requirements managers, developers, testers, customers, along with other stakeholders, e.g., project managers or quality assurance team could benefit from the application of blockchain, since it ensures full visibility on the software development

lifecycle and facilitates tracking projects' progress. Enabling full visibility can enhance the performance of practitioners in solving software engineering tasks. For instance, keeping track of all changes in a transparent and reliable manner facilitates the analysis of the impact of these changes on system's cost, time, and quality, which is not a trivial task in distributed settings.

The authors implemented the proposed BC4RT prototype using a novel neural distributed ledger, namely NDL ArcaNet because the inherent features of this platform ensure performance efficiency, sustainability and scalability while retaining security. The authors perceive the potential of third and fourth generation blockchain platforms and encourage further exploration of the benefits and feasibility of such platforms beyond the software engineering context. Domains that need to process massive data, such as Internet-of-Things (IoT) may greatly benefit from the efficiency and increased security of neural blockchain platforms.

Our future work will focus on validating the usefulness, practicality, and validity of the blockchain-enabled prototype through software engineering experts' judgement. Future versions of the prototype may incorporate the emission of tokens to represent other software artifacts, such as source code and test cases, as children of requirements' tokens. In addition, future work may be devoted to automate the registration of software artifacts, their attributes, content and changes, by means of data ingestion tools or plugins that can capture the artifacts generated from a variety of tools used throughout the software development lifecycle [32].

References

1. Ebert, C., Kuhrmann, M., Prikladnicki, R.: Global software engineering: evolution and trends. In: 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE), pp. 144–153 (2016)
2. Niazi, M., Mahmood, S., Alshayeb, M., et al.: Challenges of project management in global software development: a client-vendor analysis. *Inf. Softw. Technol.* **80**, 1–19 (2016). <https://doi.org/10.1016/j.infsof.2016.08.002>
3. Rempel, P., Mäder, P., Kuschke, T., Philippow, I.: Requirements traceability across organizational boundaries - a survey and taxonomy. In: Doerr, J., Opdahl, A.L. (eds.) *Requirements Engineering: Foundation for Software Quality*, pp. 125–140. Springer, Berlin, Heidelberg (2013)
4. Demi, S., Sanchez-Gordon, M., Colomo-Palacios, R.: What have we learnt from the challenges of (semi-) automated requirements traceability? A Discussion on blockchain applicability. *IET Softw.* **15**(6), 391–411 (2021)
5. 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)
6. 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. *Requirements Eng.* **25**(1), 21–45 (2018). <https://doi.org/10.1007/s00766-018-0306-1>
7. Zheng, Z., Xie, S., Dai, H.-N., et al.: Blockchain challenges and opportunities: a survey. *Int. J. Web Grid Serv.* **14**, 352–375 (2018)

8. Belotti, M., Božić, N., Pujolle, G., Secci, S.: A vademecum on blockchain technologies: when, which, and how. *IEEE Commun. Surv. Tutorials* **21**, 3796–3838 (2019). <https://doi.org/10.1109/COMST.2019.2928178>
9. 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>
10. 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>
11. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. *Decentral. Bus. Rev.* 21260 (2008)
12. 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
13. Marchesi, M.: Why blockchain is important for software developers, and why software engineering is important for blockchain software (Keynote). In: 2018 International Work-shop on Blockchain Oriented Software Engineering (IWBOSE), p. 1 (2018)
14. 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>
15. 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. Association for Computing Machinery, Porto, Portugal*, pp. 1–3 (2018)
16. 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
17. 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)
18. 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)
19. Damian, D., Chisan, J.: An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management. *IEEE Trans. Softw. Eng.* **32**, 433–453 (2006). <https://doi.org/10.1109/TSE.2006.61>
20. Franch, X., Fernández, D.M., Oriol, M., et al.: How do practitioners perceive the relevance of requirements engineering research? an ongoing study. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 382–387 (2017)
21. Fucci, D., Palomares, C., Franch, X., et al.: Needs and challenges for a platform to support large-scale requirements engineering: a multiple-case study. In: *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. Association for Computing Machinery, New York, NY, USA*, pp. 1–10 (2018)
22. Akbar, M.A., Sang, J., Khan, A.A., Hussain, S.: Investigation of the requirements change management challenges in the domain of global software development. *J. Softw. Evol. Process* **31**, e2207 (2019)
23. Jayatilke, S., Lai, R.: A systematic review of requirements change management. *Inf. Softw. Technol.* **93**, 163–185 (2018). <https://doi.org/10.1016/j.infsof.2017.09.004>

24. 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)
25. 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>
26. Mäder, P., Gotel, O.: Towards automated traceability maintenance. *J. Syst. Softw.* **85**, 2205–2227 (2012). <https://doi.org/10.1016/j.jss.2011.10.023>
27. Velasco, C., Colomo-Palacios, R., Cano, R.: Neural distributed ledger. *IEEE Softw.* **37**, 43–48 (2020). <https://doi.org/10.1109/MS.2020.2993370>
28. Swan, M.: Blockchain Thinking : the brain as a decentralized autonomous corporation [Commentary]. *IEEE Technol. Soc. Mag.* **34**, 41–52 (2015) .<https://doi.org/10.1109/MTS.2015.2494358>
29. Arca. In: ByEvolution Creative Factory. <https://byevolution.com/en/arca/>. Accessed 8 Jun 2022
30. iTrust. In: SourceForge. <https://sourceforge.net/projects/itrust/>. Accessed 23 May 2022
31. Meneely, A., Smith, B., Williams, L.: Appendix B: iTrust electronic health care system case study. *Softw. Syst. Traceability* 425 (2012)
32. Demi, S., Sánchez-Gordón, M., Colomo-Palacios, R.: A blockchain-enabled framework for requirements traceability. In: Yilmaz, M., Clarke, P., Messnarz, R., Reiner, M. (eds.) EuroSPI 2021. CCIS, vol. 1442, pp. 3–13. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85521-5_1



PAPER VII

S. Demi, M. Sánchez-Gordón, and R. Colomo-Palacios, "*Trustworthy and Collaborative Requirements Traceability: Validation of a blockchain-enabled framework*", Submitted to Journal of Software: Evolution and Process.