Received: 6 March 2021 | Revised: 2 June 2021 | Accepted: 17 June 2021

DOI: 10.1049/sfw2.12035

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, Østfold University College, Halden, Norway

Correspondence

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

Email: ricardo.colomo-palacios@hiof.no

Funding information

This work is part of a PhD project, funded by Østfold 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]	Kannenberg 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 Kuhrmann 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.

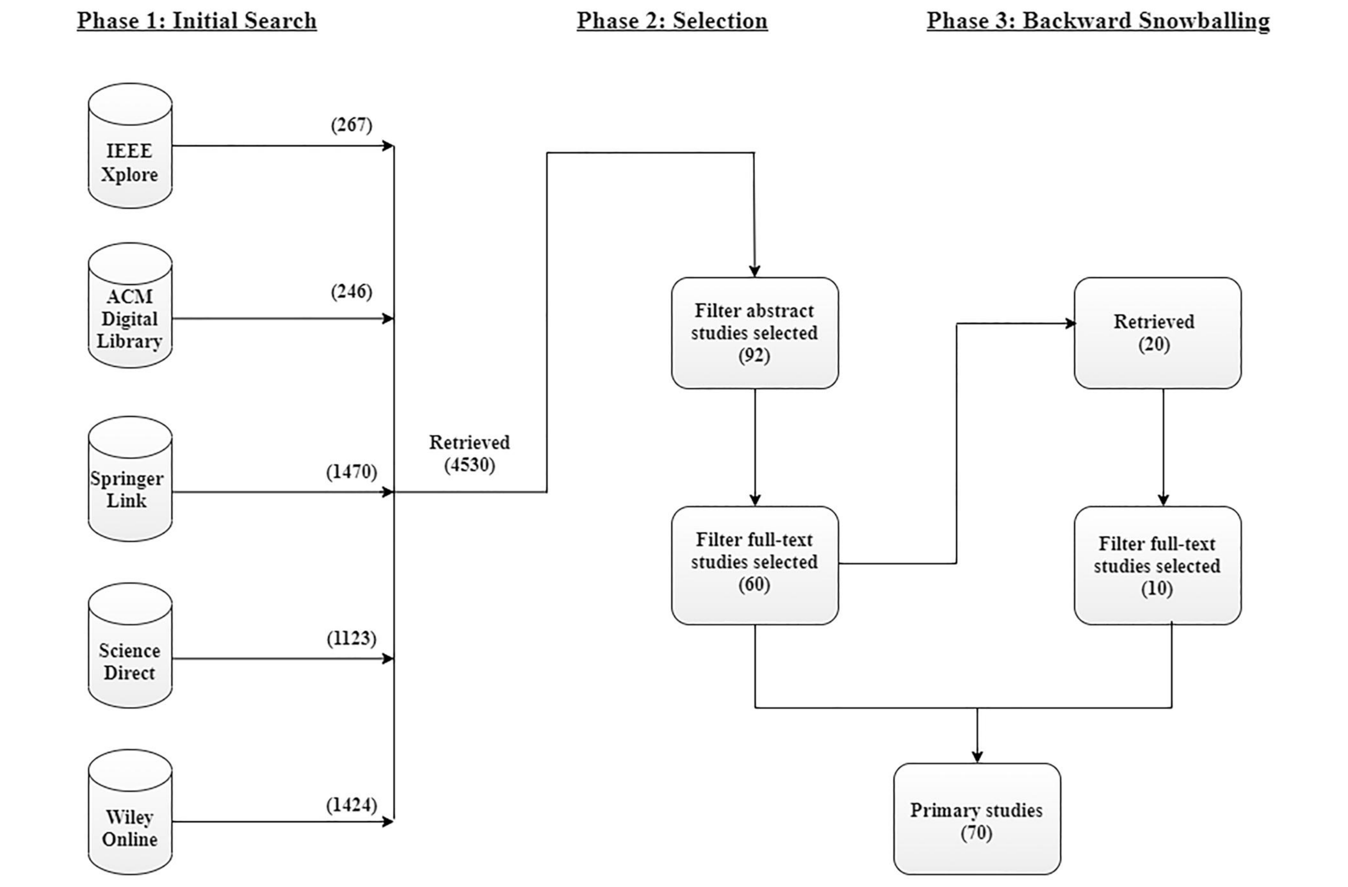

FIGURE 1 Three-stage selection process

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

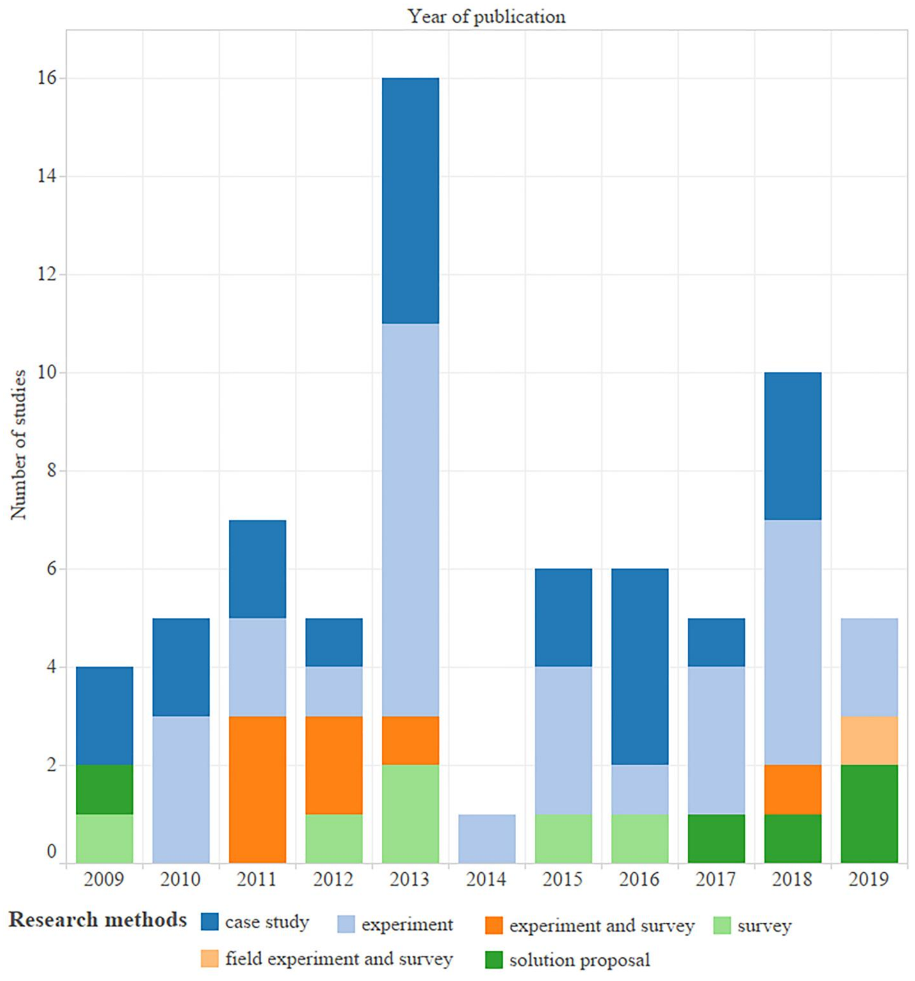

FIGURE 2 Distribution of primary studies based on publication year and research methods

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.

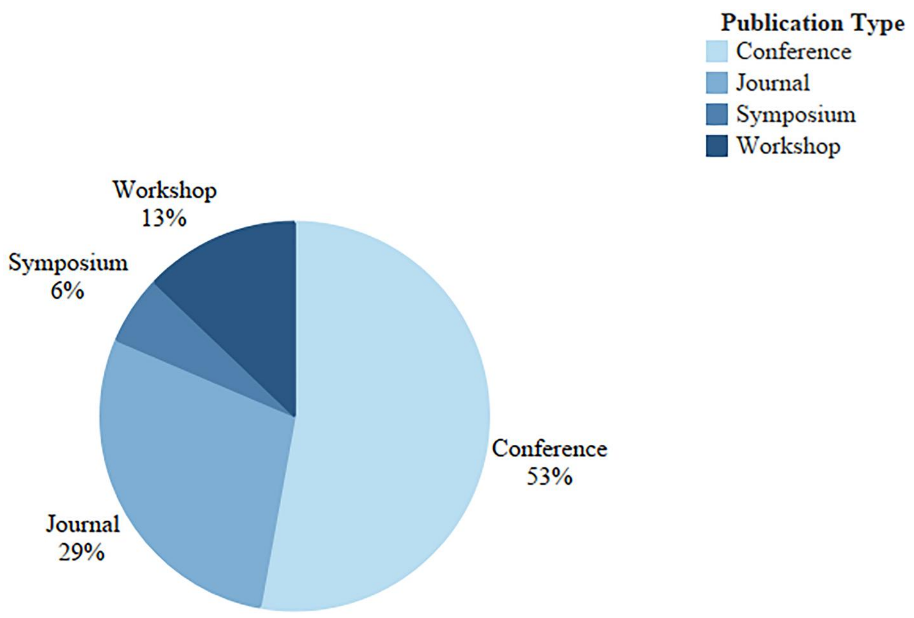

FIGURE 3 Distribution of primary studies based on publication type

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].

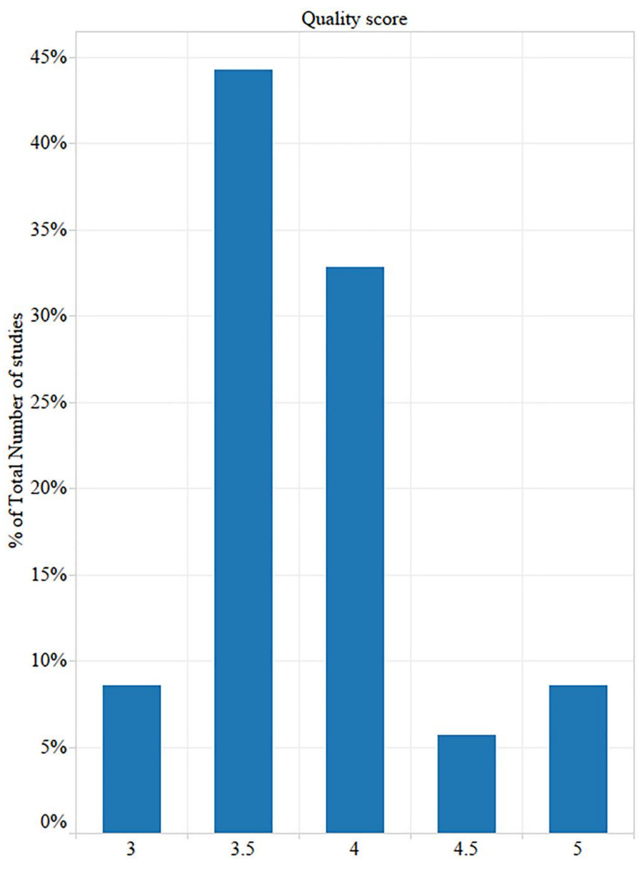

FIGURE 4 Percentage of primary studies for quality score

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]	8 (11%)
	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 heterogenous 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 Gotel [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 Gotel'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 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

1. Pohl, K.: Requirements Engineering: Fundamentals, Principles, and Techniques, 1st ed. Springer Publishing Company, Incorporated (2010)
2. 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)
3. Murugappan, S., Prabha, D.: Requirement traceability for software development lifecycle. Int. J. Sci. Eng. Res. 8(5), 1–11 (2017)
4. 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)
5. 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)
6. Kchaou, D., et al.: Recovering semantic traceability between requirements and design for change impact analysis. Innov. Syst. Softw. Eng. 15(2), 101–115 (2019)
7. 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)
8. 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)
9. Mäder, P., Olivetto, R., Marcus, A.: Empirical studies in software and systems traceability. Empir. Softw. Eng. 22(3), 963–966 (2017)
10. 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)
11. Gotel, O., et al.: Traceability fundamentals. In: Cleland-Huang, J., Gotel, O., Zisman, A. (eds.) Software and Systems Traceability, pp. 3–22. Springer (2012)
12. 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)
13. Wang, B., et al.: Requirements traceability technologies and technology transfer decision support: a systematic review. J. Syst. Softw. 146, 59–79 (2018)
14. Sundaram, S.v et al.: Assessing traceability of software engineering artifacts. Requirements Eng. 15(3), 313–335 (2010)
15. 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)
16. 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)
17. Winkler, S., von Pilgrim, J.: A survey of traceability in requirements engineering and model-driven development. Softw. Syst. Model. 9(4), 529–565 (2010)
18. Blaauboer, F., Sikkel, 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)
19. 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)
20. 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)
21. Saiedian, H., Kannenberg, A., Morozov, S.: A streamlined, cost-effective database approach to manage requirements traceability. Softw. Qual. J. 21(1), 23–38 (2013)
22. Niu, N., Wang, W., Gupta, A.: Grey links in the use of requirementstraceability. In: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 384–395. ACM (2016)
23. Gayer, S., et al.: Lightweight traceability for the agile architect. Computer. 49(5), 64–71 (2016)
24. Fernández, D.M.: Supporting requirements – engineering research that industry needs: The NaPiRE Initiative. IEEE Softw. 35(1), 112–116 (2018)
25. 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)
26. Torkar, R., et al.: Requirements traceability: a systematic review and industry case study. Int. J. Softw. Eng. Knowl. Eng. 22, 385–433 (2012)
27. Maro, S., Steghöfer, J.-P., Staron, M.: Software traceability in the automotive domain: challenges and solutions. J. Syst. Soft. 141, 85–110 (2018)
28. Gotel, O., et al.: The Grand Challenge of Traceability (v1. 0) (2011)
29. Ramesh, B.: Factors influencing requirements traceability practice. Commun. ACM. 41(12), 37–44 (1998)
30. Kannenberg, A., Saiedian, H.: Why software requirements traceability remains a challenge. J. Defense Softw. Eng. 22, 14–19 (2009)
31. Nair, S., de la Vara, J.L., Sen, S.: A review of traceability research at the requirements engineering conferencere@21. In: 2013 21st IEEE International Requirements Engineering Conference (RE), pp. 222–229. (2013)
32. Wieringa, R., et al.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirements Eng. 11(1), 102–107 (2006)
33. Kitchenham, B.: Procedures for Performing Systematic Reviews, p. 33
34. 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)
35. 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)
36. 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)
37. 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)
38. 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)
39. What have we learned from the challenges of (semi-) automated requirements traceability? A discussion on blockchain applicability. figshare (2020)
40. 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)
41. 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)
42. 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)
43. 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., Sjøberg, 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., Mçder, 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. 1–21 (2021). https://doi.org/10.1049/sfw2.12035

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