Requirements Engineering (2023) 28:619–637
https://doi.org/10.1007/s00766-023-00408-9

ORIGINAL ARTICLE

Why don't we trace? A study on the barriers to software traceability in practice

Marcela Ruiz[1] · Jin Yang Hu[2] · Fabiano Dalpiaz[2]

Received: 22 June 2022 / Accepted: 24 October 2023 / Published online: 30 November 2023

Abstract
Researchers have proposed numerous tools, methods, and techniques for establishing and maintaining software traceability. Despite its acknowledged importance, researchers argue that traceability is still "a sought-after, yet often elusive quality in software-intensive systems". We have little evidence regarding how creating, managing, and using traceability links vary depending on factors such as organizational contexts, software development practices, and project types. We conduct an empirical study where software development practitioners express their perception regarding the value of software traceability. Via an online survey, 55 participants provided information related to their current traceability practices and needs. Furthermore, we interviewed 14 practitioners to gain a more in-depth understanding. Our study investigates the effect of two independent variables: the software development paradigm and the type of developed software system. Among the several identified findings, our analysis reveals that, although the traceability costs are an inhibitor for adopting more mature traceability practices, the respondents believe that the expected benefits still outweigh envisioned costs. Traceability is mainly performed manually: not only are automated trace retrieval tools scarce, but their offered automation is not expected to replace human involvement.

Keywords Software traceability · Exploratory study · Agile development · Safety-critical systems

1 Introduction

Academia has dedicated significant attention to software traceability over the past decades. Traceability researchers have identified several challenges that, if solved, are expected to make traceability as effortless as possible (*ubiquitous traceability*) for software engineers to establish and maintain [1, 16, 17].

Much of the traceability research focuses on the construction and evaluation of *automated techniques* for the establishment, maintenance, and recovery of trace links between requirements, code, and other software artifacts [9, 12, 20, 37]. One of the primary reasons for this focus on automation is that manual traceability requires considerable human effort, which often leads to trace links that quickly become outdated [23].

In their roadmap paper, Cleland-Huang et al. [10] argue that traceability receives more attention in regulated and safety-critical domains, while in other areas practitioners use it less often or are even unaware of the term. Moreover, Gotel et al. observe the prominence of ad-hoc traceability methods, which are however unable to fully support the evolving stakeholders' needs [16, 17]. Regarding development teams that adopt Agile methods, there is little evidence of use cases and challenges, despite the call for studies to investigate this setting [7].

As such, there is still a need to expand the body of knowledge on the usage of software traceability in the software industry and the benefits and limitations that practitioners face when implementing and maintaining traceability. This imbalance between theoretical and empirical research defines our **main research goal** (MRG): *gaining contemporary empirical knowledge on software traceability in terms of stakeholder perceptions, their current practice, their challenges, and their needs.* Specifically, we wish to uncover whether the *perceived benefits and challenges* regarding

✉ Marcela Ruiz
marcela.ruiz@zhaw.ch

[1] Zürich University of Applied Sciences, Switzerland, Technikumstrasse 9, Winterthur 8401, Switzerland

[2] Utrecht University, Utrecht 3584 CC, The Netherlands

software traceability are affected by (i) the software development paradigm (agile vs. traditional development) and (ii) the safety-critical nature of the systems under design. These two variables were not examined by previous studies on how traceability is perceived in practice.

We investigate the perceived cost-benefit ratio: in domains where software traceability is not mandated by regulations, the reasons behind the adoption of traceability and the selection of a specific method are dependent on whether practitioners perceive traceability to deliver value [2] and whether the perceived benefits outweigh the expected costs [5].

To gain contemporary knowledge on these topics, we started from existing literature and we assembled a questionnaire that we distributed through an online survey. We received answers from 55 participants from 13 different countries that provide information regarding their current traceability practices and their needs. We complemented and contextualized the results of the survey by conducting in-depth interviews with 14 practitioners (8 of them were respondents to the survey), which allowed us to obtain a more nuanced understanding of their practices and perception on traceability.

Our analysis results into a set of findings (9 from the survey, 8 from the interviews) that support both researchers and practitioners by delivering an *updated picture of software traceability* practices and needs. Researchers can use our results to better direct their research efforts into methods that can actually support practitioners. Practitioners can relate their own situation with that of their peers and decide whether adopting different practices could be beneficial.

Paper organization. After reviewing related work in Sect. 2, we describe the research method in Sect. 3. Section 4 presents the results from the survey, while Sect. 5 details the findings from the interviews. Section 6 presents conclusions, discussed validity threats, and outlines future work.

2 Studies on traceability in practice

While the community keeps proposing new, intelligent approaches for establishing and maintaining traceability [15, 21, 24, 30, 32, 37], fewer studies address traceability from the perspective of practitioners. However, the results of these studies are of paramount importance, for much of the traceability research is driven by practitioners' needs and their involvement [3, 17, 22].

A precursory study was conducted by Gotel and Finkelstein in the mid-1990s [18]. Through focus groups and a survey, they investigated the underlying nature of the requirements traceability problem. Their work introduces the distinction between pre-requirements specification traceability and the post-requirements specification traceability. Their results show that poor traceability practices are often due to tool limitations and to weak collaboration between end-users and providers. Another identified issue is the inability to locate and access the sources of requirements. This problem is worsened when the end-users are split across multiple teams; factors that reduce the problem, instead, include small teams, clear separation of responsibilities, team commitment and ownership.

In the late 1990s, Ramesh [31] studied how environmental, organizational, and system development factors influence the adoption and use of requirements traceability. An important distinction that emerged was between low-end and high-end users. The former see traceability as a mandate and implement simple traceability tactics. High-end users, instead, see traceability as an important component of their development and use more advanced tactics. Ramesh found an alignment between end users and their managers: the managers of low-end users claim that little benefit is obtained and see traceability as 'necessary evil'. In contrast, the managers of high-end users are committed to traceability and see strategic benefits of incorporating traceability practices, even when not mandated.

In 2005, Arkley and Riddle surveyed nine software projects of varying complexity [2], and their findings are aligned with those from previous studies. They argue that poor traceability practices are due to the traceability benefit problem: the lack of direct, tangible benefits, which contribute to seeing traceability as an overhead. They explicitly list *perceived benefit* as an important cause for the lack of adoption. These findings align with the conclusions from the case study reported by Maro et al. [28] pointing out at the lack of explicit traceability return of investment as a major barrier for using traceability in practice.

Blaauboer et al. conducted a case study in a large IT company to identify factors that affect traceability adoption in information systems development projects [5]. Five dominant factors emerged: development organization awareness, customer awareness, return on investment, stakeholder preferences, and process flow. When project leaders were unaware of the notion of traceability, trace links were not even considered.

In 2009, Mäder et al. conducted a practitioner survey to obtain up-to-date evidence on traceability practices and problems [26]. Their major findings included that traceability is necessary, but almost no guidance was available, especially for trace links that span across project, organizational, and regional boundaries. The role of tools was observed to have become more important in practice, probably due to the increasing complexity of the systems under design.

Bouillon et al.'s survey [6] identified a number of frequent usage scenarios in practice, which included finding origin and rationale of requirements, documenting a requirement's history, and tracking requirement or task implementation state. Nonetheless, the authors emphasized that very little is known about practitioners' uses and needs, and they called for more studies to shed light on the traceability benefit problem.

Wohlrab et al. conducted a multiple case study across 15 industrial projects [39]. They analyzed the collaborative aspects of traceability from the perspectives of organization, process, and culture. The findings show that practitioners struggled with (1) collaboration across team and tool boundaries, (2) conveying the benefits, and (3) traceability maintenance.

These studies reveal many factors that influence the adoption and use of software traceability. Regan et al. [33] organized these factors into a taxonomy of the barriers of traceability in practice and their potential solutions. Although their study dates back to 2012, recent studies show a significant overlap (see, e.g., [39]). Because of its structure and general perspective on traceability, which encompasses managerial, social, and technical issues, we employ their taxonomy of barriers (see Table 1) as the baseline for our investigation.

Traceability was shown to have benefits: the experiment of Mäder and Egyed [25], for example, evidences how it can positively impact software engineering tasks. Their controlled experiment with over 50 subjects performing maintenance tasks shows that a simple trace navigation tool has a positive effect on the performance, quality, and workflow of how change tasks are performed. Our research aims to shed some light on the *value* (benefit minus cost) and barriers that practitioners perceive. Aligned with the observations by prominent researchers in the field [8, 17], we include not only the 'usual' safety-critical and heavy regulated industries, but focus also on more recent paradigms such as agile.

Table 1 Barriers of traceability by Regan et al. [33]

Category	Barriers
Management	Cost
	Lack of guidance
	Return on investment
	Traceability decay
	Data collection
Social	Different stakeholder viewpoints
	Internal politics
	Lack of communication/understanding
Technical	Issues with tools
	Storage and versioning
	Complexity

3 Study definition and planning

The following sub-section describe our study setting according to the guidelines by Wohlin and colleagues [38].

3.1 Main research goal

Our main research goal, stated in the introduction, focuses on *gaining contemporary knowledge on software traceability from practitioners, and to analyze the effect of contextual factors such as the used software development paradigm, and whether the developed software is safety critical.* To achieve this goal, we have conducted an online questionnaire, followed by semi-structured interviews to gain deeper insights. We applied the protocols described by Robson et al. [34].

Both questionnaire and interviews are designed to gather practitioners' perceptions about the categories presented in Table 1 [33]: *management*, *social*, and *technical*. We have derived statements that encourage practitioners to express their perceptions in a free manner, without any bias induced by the researchers' opinion (in favor or against certain aspects of traceability). Each category is investigated in terms of practitioners' perceived barriers accounting both practitioners' *current traceability practices* and *needs*. Finally, during the interviews, we also examine the role of traceability in the software development life cycle (SLDC).

3.2 Research questions and variables

The research questions we have formulated to study the perceived relevance of the barriers to software traceability in practice are as follows:

RQ1: *Which is the perceived importance of the barriers to software traceability in practice?*
To answer this research question, we gather practitioners' perceptions according to the categories identified by [33].
RQ2: *Is the software development paradigm influencing the perceived importance of barriers to software traceability?*
RQ3: *Is the type of software system influencing the perceived importance of barriers to software traceability?*

The second and third research questions relate to the following two independent variables:

Software development paradigm. The paradigm that guides the way practitioners develop software. This variable has two values:

- *Agile*. This value groups methods that build on the agile manifesto [4].
- *Traditional*. Methods that build mostly on traditional software development paradigms, e.g., the V-model [13].

Type of software system. The type of software system that practitioners work on can assume two values:

- *Safety-critical*. Systems whose failure may harm human or environmental safety [36].
- *Non-safety-critical*. Systems whose malfunctions are not critical.

We consider the following dependent variables, which are expected to be influenced by the independent variables. We made use of the categories stated for the barriers of traceability identified by Regan and colleagues [33] to structure the way the dependent variable is measured. In this way, the perceived importance of the barriers to software traceability is measured by evaluating practitioners' current traceability practices and needs to establishing traceability (see Fig. 1).

Perceived barriers in current traceability practices. The degree to which practitioners agree with the barriers to software traceability by considering their current software traceability practices. This variable is measured in the context of the management, social, and technical aspects suggested by the categories of traceability by Regan et al. [33]. We measure their agreement with the corresponding statements via Likert-type answers: "Strongly disagree", "somewhat disagree", "somewhat agree", and "strongly agree".

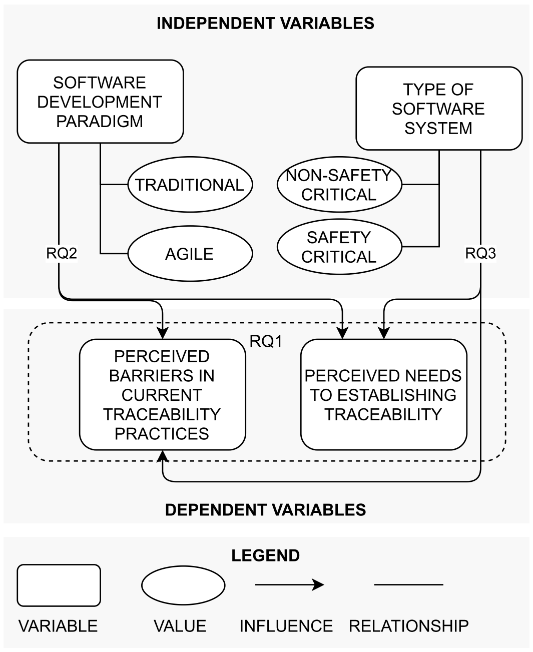

Fig. 1 Structure of the independent and dependent variables in our research

Perceived needs to establishing traceability practices. The practitioners' priority on the need for software traceability for management, social, and technical aspects suggested by the categories of traceability by Regan et al. [33]. Their priority for the corresponding statements is measured via Likert-type answers: "not needed", "nice to have", "should have", and "must have".

3.3 Study design

Questionnaire. We created questions and statements that would allow us to answer RQ1–RQ3. After a pilot with six master's students in computer science, we interacted with a native UK English speaker to validate the formulation of the questions and with a software developer to validate the content further. The pilot led to some clarifications and an improved layout. The pilot also helped us to ensure the questionnaire could be quickly filled in by the participants (10–15 min).

The participants are first presented with the front page which explains the goal of the questionnaire. After that, to establish a common ground, an illustrative figure is shown to provide a general definition of software traceability. The first questions gather demographic data about the participants, their organization, and their project characteristics. Then, 13 statements are shown regarding their current or most recent practices using the Likert-type answers to measure the first dependent variable. The following 13 statements focus on prioritizing their needs, and allow us to measure the second dependent variable. Finally, the last page allows to provide additional feedback about the questionnaire or the topic, and to describe more of their rationale where needed. The questionnaire is included in our online appendix [35], and the main questions are listed in Table 2. All the statements about the current practices, except for one, are written in such a way that strong agreement indicates that an obstacle is perceived as very important. The exception is C6-S: to facilitate reading, the obstacle of "low perceived importance of traceability" is reversed into the statement "Traceability is of high importance for the software development process". Therefore, the responses are inverted in our analysis: strong agreement with the statement indicates strong disagreement with the importance of the obstacle.

Semi-structured interviews. In order to gain more in-depth, contextual information, we conduct interviews with practitioners: the interview protocol follows the same questions of the questionnaire, but it encourages participants to explain the rationale behind each answer. Moreover, we include one additional question on which software

Table 2 Statements from our questionnaire, inquiring about the current situation and the needs from the managerial, social, and technical perspective

Current Situation (strongly disagree, somewhat disagree, somewhat agree, strongly agree)	
C1-M	Traceability costs in money, time, and effort are the main inhibitor for adopting (more mature) traceability practices
C2-M	There is insufficient guidance within the company on how to establish traceability
C3-M	Traceability costs outweigh the expected benefits
C4-M	It is difficult to access and obtain information sources (people and artifacts) to be able to establish traceability
C5-M	Traceability is mostly performed in an ad-hoc and non-managed fashion
C6-S*	Traceability is of high importance for the software development process
C7-S	The allocation of time, staff and resources are often insufficient to be able to properly establish and maintain traceability
C8-S	There is a lack of collaboration between involved stakeholder teams in regards to establishing and maintaining traceability
C9-S	It is unclear which roles are responsible for traceability
C10-T	Traceability tools do not satisfy our traceability needs
C11-T	It is difficult to establish traceability because development artifacts are stored in multiple locations/repositories
C12-T	The tools used for traceability are too complex to use effectively and to integrate with our existing tools
C13-T	Traceability is mostly performed manually
Needs (not needed, nice to have, should have, must have)	
There is a need ...	
N1-M	. . . to reduce the costs of traceability
N2-M	...for more guidance in the company regarding traceability
N3-M	...to have a more clear overview of the costs and benefits regarding traceability
N4-M	...to have easier access to information sources to be able to establish and maintain traceability
N5-M	...to perform traceability in a more managed fashion
N6-S	...to increase the awareness of the importance of traceability
N7-S	...for more staff, time, and resources to properly establish and maintain traceability
N8-S	...for more collaboration between involved stakeholder teams regarding traceability
N9-S	...for more communication about who are responsible for traceability
N10-T	...for better traceability tools
N11-T	...for a more centralized development artifact repository to establish traceability more easily
N12-T	...for less complex traceability tools and easier integration with our existing tools
N13-T	...for more traceability automation

The perspective is indicated by the suffix: e.g., ‘C1-M’ is a managerial challenge. Question Q6-S has a reversed scale compared to the other questions: the higher, the fewer obstacles

development life cycle phase is the one where software traceability gives the highest value. The interviewer first introduces the goal of the interview, then makes questions on the professional background and the organizational background. The same flow of the questionnaire is followed (to explore current traceability practices, needs), with the difference that the focus is on uncovering the rationale and that, to keep a natural flow, we allow covering the answers without rigidly imposing the flow. The results from the interview were analyzed following the questions of the questionnaire. The topic of the questions allowed for identification of findings. We explored the interview transcripts in order to show evidence on the findings and conclusions. Further details on the results can be found in Sect. 5, and the interview protocol can be accessed in the online appendix.

3.4 Selection of participants and demographics

Our target participants consist of a broad range of practitioners with experience in software development with various roles, including product owner, requirements analyst, developer, and tester. Considering it is mainly and opinion-based study, we did not exclude participants who do not perform traceability in their projects. The participants were mainly contacted via LinkedIn, Reddit, and e-mail. In total, our study is based on data from 61 practitioners located in 15 different countries. In particular, 55 practitioners participated in the questionnaire, 8 of which volunteered to be interviewed. Also, we interviewed 6 additional practitioners to obtain a richer landscape, reaching a total of 14 interviews (circa 1-h each). The questionnaire data from the 6 extra interviews are not included in the survey results to avoid construct validity threats caused by the possible bias introduced by the interaction with the researchers during the interviews. An overview of the demographic data is shown in Fig. 2.

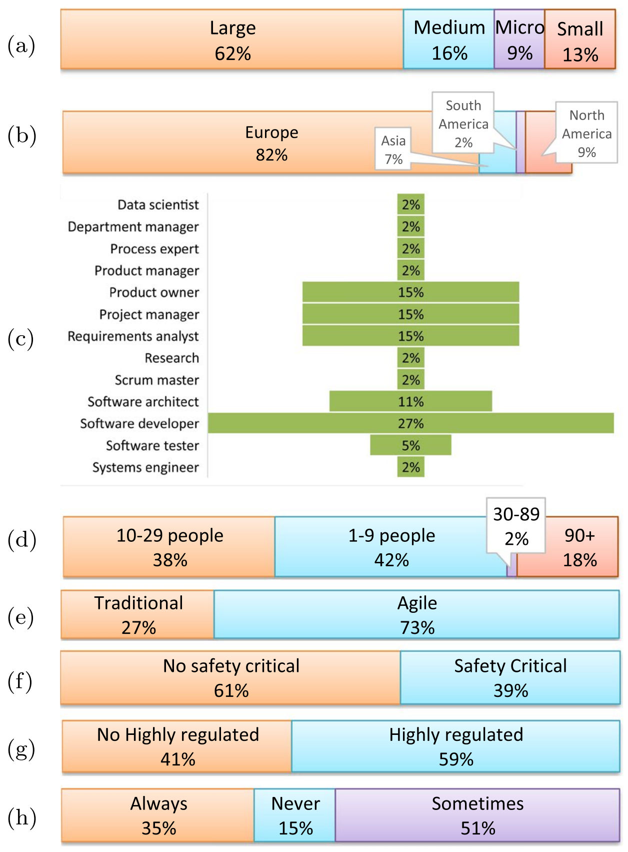

Fig. 2 Distribution of the 61 practitioners by **a** organization size, **b** continent, **c** role, **d** team size, **e** software development paradigm, **f** type of software system, **g** highly regulated, **h** frequency traceability is performed

4 Survey results

In the following three sub-sections, we report our results per each research question (RQ1–RQ3). In particular, we present key findings and we elaborate on the corresponding evidence.

Before analyzing the barriers, we first investigate the reasons expressed by the practitioners for adopting traceability. This was a multiple choice question based on Bouillon et al.'s study [6]. Four non-exclusive choices were offered to practitioners, as shown in Table 3: expected benefit, mandate, customer request, or other. From the original taxonomy, we removed the choice 'precondition for applied method or tool', since it is partially linked to our independent variable 'software development paradigm'. The data in Table 3 are based on the 47 out of 55 respondents who said they use traceability *sometimes* or *always*: the participants who said they are currently *not employing* traceability were not asked for the reasons for adopting traceability. The columns of the table indicate the reason, the percentage of respondents who selected a reason, a split-down of the responses depending on whether the participants indicated they follow agile methods, a similar split-down based on the self-reported safety-criticality, and the results from Bouillon's study.

Then, we ran Pearson's χ^2 test, which is appropriate to analyze differences between groups of categorical data, between the results from our study and those in Bouillon's [6]. We also tested whether the differences between groups (agile vs. non-agile, safety-critical vs. non-safety-critical) are statistically significant. The results from the statistical tests are shown in Table 4.

Finding S1. The reasons for adopting traceability are similar to those in Bouillon et al.'s study [6]. However, fewer respondents indicated the expected benefits as a reason for embracing traceability.

Evidence for S1. The results in Table 3 show that expected benefit and mandate are by far the prevalent reasons: 57% and 48%, respectively. The percentage for the benefit, however, decreased considerably from Bouillon's study, where 80% of the practitioners indicated that as a

Table 3 Reasons for adopting traceability; the respondents could express multiple options, as per the original study [6]

Reason	Our study					2013 [6]
	Total	Agile		Safety-critical		
		Yes	No	Yes	No	
Expected benefit	57.4	51.6	68.7	47.0	66.3	80
Mandate	48.9	45.1	56.2	64.7	40.0	39
Customer request	27.7	22.5	37.5	47.0	16.6	30
Other reasons	19.2	19.3	18.7	17.6	20.0	2

The numbers are percentages

Table 4 Testing for the existence of statistically significant differences between groups: (i) our study versus Bouillon's, total responses; (ii) our study, traditional versus agile; and (iii) our study, safety-critical versus non-safety-critical

Reason	Pearson χ^2 Test	
	χ^2	p
Our study versus Bouillon		
Expected benefit	6.375	0.012
Mandate	0.967	0.325
Customer request	0.090	0.764
Tradit versus Agile		
Expected benefit	1.268	0.260
Mandate	0.519	0.471
Customer request	1.174	0.279
Safety-crit versus Non-SC		
Expected benefit	1.176	0.278
Mandate	2.651	0.104
Customer request	5.009	0.025

reason. Such a difference is statistically significant with a p value < 0.05 resulting from the χ^2 test. In our sample, the expected benefit is a more common reason for the practitioners who use non-agile development paradigms (68.7% vs. 51.6%). Additionally, we have analyzed if there is significant difference in the results between the groups in our study: Traditional versus Agile, Safety-Critical versus Non-Safety-Critical for each of the reasons (Table 4). In the context of safety-critical systems, there is a significant difference between the groups who applied traceability because of customer request versus the group in the context of non-safety-critical systems. This result highlights the value of traceability demanded by customer request given the criticality of the software system under development. This outcome may lead to further research regarding the reasons for adopting traceability.

4.1 General analysis of the obstacles (RQ1)

To answer RQ1, we present findings that can be derived from the aggregate answers to the statements listed in Table 2, before splitting the data according to the independent variables. Our investigation focuses on the two main dependent variables of perceived barriers of software traceability: current practices and needs.

Figures 3 and 4 depict the results of the statements regarding the respondents' current traceability practices and their needs. The short labels in the figures can be traced back to the actual questions in Table 2 via the unique identifier (e.g., C1-M, N10-T, ...). We use divergent stacked bar charts for visualizing the results from our Likert-type questions. These

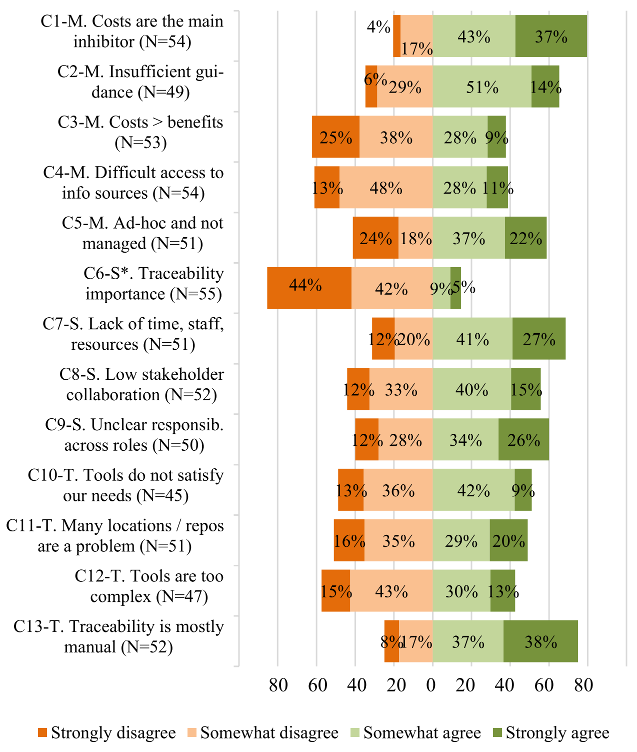

Fig. 3 Results regarding the statements on the current practices, including the number of responses per question. The results for C6-S are shown reversed because of the question's formulation, as explained in the text

charts provide a quick overview of the trends and differences between groups based on the percentage of the participants who chose a certain answer. Each bar is evenly sized and represents the entire population; a single bar relates to one statement, and it is divided into portions, each denoting the percentage of one answer, like a typical bar chart. The bars are centered around a zero line, which separates the 'positive' from the 'negative' answers.

As explained in Sect. 3, C6-S is formulated in positive terms ("traceability is of high importance"), rather than as an obstacle ("traceability is of low importance"); therefore, the results are reversed in the charts, for consistency with the other statements.

Figure 3 shows the statements regarding the current practices (C1-M through C13-T, from Table 2), each answered by a subset of the 55 respondents. While for some statements a key prevalence is visible (e.g., C6-S on the high importance of traceability), for others no clear divergence is visible (e.g., C10-T: tools do not satisfy our needs). Nevertheless, both situations are interesting, given that each of our statements represents one of the *barriers to traceability* identified by Regan et al. [33], listed in Table 1.

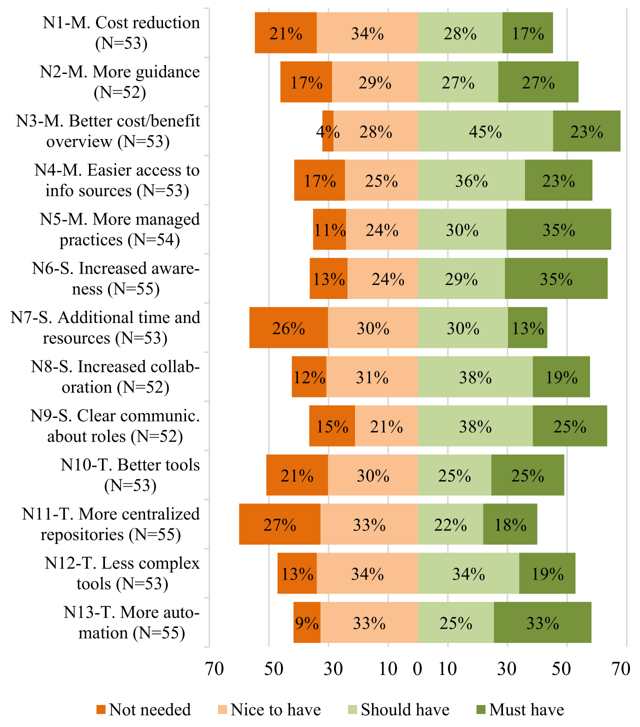

Fig. 4 Results for the statements on the expressed needs

Finding S2. Although costs seem to be a key inhibitor to traceability, the respondents do not find these costs to outweigh the benefits.

Evidence for S2. The answers to C1-M indicate that 80% of the respondents perceive costs as the main inhibitor to traceability (see Fig. 3). Nevertheless, when we cross-check this with the answers to C3-M, we see that only 37% of the respondents believe that costs outweigh the benefits (with only 9% of them strongly agreeing with the statement).

Finding S3. Traceability is of high importance for the software development life cycle.

Evidence for S3. The answers to C6-S (reversed in the figure) on the high importance of traceability are aligned: 86% of the respondents agree, either strongly or weakly, with an even distribution of these two levels of agreement (see Fig. 3). This sentiment is confirmed by the answers to C3-M: the costs of traceability do not outweigh the benefits.

Finding S4. Contrary to prior evidence, tool complexity and difficult access to information sources seem to not represent a challenge.

Evidence for S4. Two of the challenges from Regan et al.'s study [33] do not seem to hold, according to our respondents. These challenges are the difficult access to information sources (see Fig. 3, C4-M, 61% disagreement, only 11% strongly agree), and the complexity of current tools (see Fig. 3, C12-T, 58% disagree, with 77% of the responses being either somewhat negative or somewhat positive). We say 'seem' to denote that this finding requires additional exploration, which we report through our interviews analysis in Sect. 5.

Figure 4 presents the results for the statements regarding the needs to change (N1-M through N13-T in Table 2). Due to the different type of scale, which is based on the priority of the need, we have decided to group 'not needed' and 'nice to have' as a negative sentiment, for 'nice to have' also indicates a weak necessity that is unlikely to lead to changes in the traceability practices.

Finding S5. Traceability has to be better embedded in the organization by improving on awareness, cost/benefit overview, roles and responsibilities, and managed practices.

Evidence for S5. Several of the statements that are often indicated as 'must have' or 'should have' relate to the embedding of traceability within the organization, i.e., they concern mostly managerial aspects. N3-M and N6-S denote how the respondents seem to want a better overview of benefits/costs (68%) and an increased awareness (64%). Furthermore, N5-M and N9-S call for better organizational practices by showing the need of more managed practices (65%) and of clear communication about roles and responsibilities (63%) (see Fig. 4).

We have also compared the three main categories of barriers (Managerial, Social, and Technical), with the aim of identifying whether a statistically significant difference would exist across the categories. Such a difference could indicate that certain types of barriers are perceived as more important (for the current practices), or that certain needs are more urgent. To do so, since we employ Likert-type data, we ran a nonparametric test that compares the population by their ranks (rather than the absolute values): we opted for Mann–Whitney's U test. We tested the three combinations: M-S, M-T, and S-T. In no case we obtained statistically significant results: the p values for the current practice are 0.14 (M-S), 0.87 (M-T), and 0.12 (S-T), while those for the needs are 0.60 (M-S), 0.43 (M-T), and 0.21 (S-T). The full results can be found in our online appendix (https://zenodo.org/records/8021723).

Finding S6. There does not seem to be a difference between the categories defined by Bouillon (managerial, social, technical), neither in terms of the current situation, nor of the perceived needs.

Evidence for S6. The statistical results listed above do not exhibit any statistically significant difference. This makes us conclude that, provided that the list of obstacles by Bouillon et al. [6] is representative for each of the categories, there is no high-level perspective from which traceability is more problematic. See the threats to validity for further discussion on this finding.

4.2 Development paradigm—RQ2

Figure 5a compares the current practices by showing side-by-side the responses of the practitioners who reported to employ traditional methods (T) and those who use agile methods (A), and Fig. 5b shows respondents opinions from the perspectives on their needs to change.

In order to test whether the differences are statistically significant, since the answers are ordinal, we employ non-parametric methods that are based on ranks and that do not require the data to be roughly normally distributed. In particular, we use Mann–Whitney's U test [27] to assess whether the difference is significant. Furthermore, we report the estimated effect size from U's test, as suggested by Fritz et al. [14], by calculating Cohen's r via the equation $r = z/\sqrt{N}$. For the qualitative interpretation of effect size, we follow Cohen's guidelines [11]: $r \in [0.1, 0.3)$ indicates a *small* effect, $r \in [0.1, 0.5)$ denotes an *intermediate* effect, and $r \geq 0.5$ represents a *large* effect.

The outcomes of the statistical analysis for the traditional versus agile development comparison are shown in Table 5. The results allow us to derive one additional finding, based on the statement that gained statistical significance and medium effect size: C6-S.

Finding S7. Although of high importance in both cases, traceability is perceived of higher importance in traditional environments than in agile developments.

Evidence for S7. All of the respondents to C6-S in the traditional development group answered that traceability is of high importance, either by somewhat agreeing (39%) or by strongly agreeing (61%). For agile development, this percentage drops to 78%. This figure is still one of the highest ones among all statements, thereby demonstrating that,

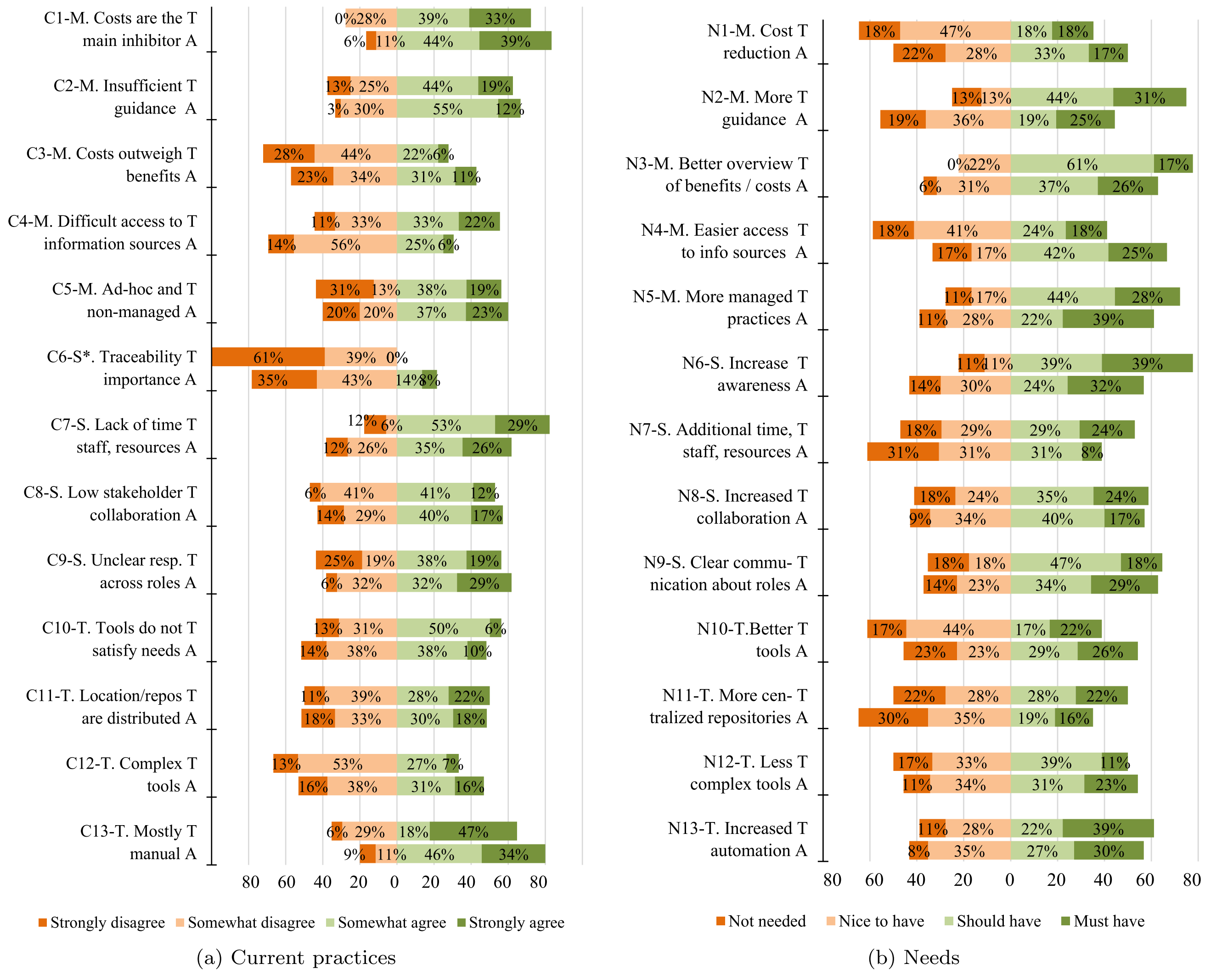

Fig. 5 Comparison of the results, splitting by the 'Software development paradigm' independent variable: **T**raditional (n = 18) versus **A**gile (n = 37)

Table 5 Statistical significance and effect size for the traditional versus agile comparison

Stat.	U	z	N	*p*	r	Effect
C1-M	292	− 0.63	54	0.528	− 0.086	–
C2-M	258	− 0.139	49	0.889	− 0.02	–
C3-M	267	− 0.946	53	0.344	− 0.13	Small
C4-M	236	− 1.737	54	0.082	− 0.236	Small
C5-M	254.5	− 0.54	51	0.589	− 0.076	–
C6-S	**218.5**	**− 2.237**	**55**	**0.025**	**− 0.302**	**Med**
C7-S	247	− 0.884	51	0.377	− 0.124	Small
C8-S	290.5	− 0.144	52	0.885	− 0.02	–
C9-S	224.5	− 1.03	50	0.303	− 0.146	Small
C10-T	221	− 0.278	45	0.781	− 0.042	–
C11-T	277.5	− 0.401	51	0.689	− 0.056	–
C12-T	210.5	− 0.713	47	0.476	− 0.104	Small
C13-T	291.5	− 0.124	52	0.901	− 0.017	–
N1-M	287	− 0.377	53	0.707	− 0.052	–
N2-M	220	− 1.396	52	0.163	− 0.194	Small
N3-M	294	− 0.422	53	0.673	− 0.058	–
N4-M	244.5	− 1.22	53	0.223	− 0.168	Small
N5-M	320	− 0.077	54	0.939	− 0.01	–
N6-S	279.5	− 1.001	55	0.317	− 0.135	Small
N7-S	237.5	− 1.357	53	0.175	− 0.186	Small
N8-S	294.5	− 0.061	52	0.951	− 0.009	–
N9-S	276.5	− 0.428	52	0.668	− 0.059	–
N10-T	293	− 0.427	53	0.669	− 0.059	–
N11-T	283	− 0.931	55	0.352	− 0.125	Small
N12-T	278.5	− 0.717	53	0.473	− 0.099	–
N13-T	311	− 0.413	55	0.68	− 0.056	–

The columns indicate the statement identifier, Mann–Whitney's U value, the z value, the number of responses N, the p value, the effect size r, and its qualitative interpretation. The rows denoting statements with statistical significance ($p < 0.05$) between groups are colored in bold

although to a lesser extent, traceability is perceived of high importance also in agile development (see Table 5).

No other statistically significant difference can be found regarding the other statements. Nevertheless, the responses to C4-M show a visually noticeable difference between the two development paradigms: for traditional development, 50% of the respondents found access to information sources to be a difficulty; for agile development, this percentage dropped to 31%, with only 6% strongly agreeing. In other words, 69% of the respondents do not find this an obstacle in agile development. This aligns well with the principles from the agile manifesto [4], in which communication between team members is emphasized.

Even weaker differences in Fig. 5a exist on the cost/benefit ratio (C3-M): traditional practitioners more strongly disagree that the costs outweigh the benefits; the high importance of traceability (C6-S): all traditional practitioners agreed, either weakly or strongly; the lack of time and resources as an obstacle (C7-S): less of a problem for agile practitioners; and that traceability is mostly manual (C13-T): stronger agreement for agile practitioners. However, the divergence is not so evident for us to identify a finding.

Finding S8. Although not perceived as an issue in the current practices, agile practitioners state that they need an easier access to information sources.

Evidence for S8. The responses to N4-M show a large divergence between the two groups. 77% of the agile practitioners find that having easier access to information sources is either a must have (25%) or a should have (42%). Quite interestingly, the corresponding statement regarding practices (C4-M in Fig. 5a) shows that only 31% of them found that it is difficult to access these sources.

When examining the needs, we cannot find statistically significant differences. We make, however, some observations that do not lead to findings. Increasing awareness of traceability seems to be more important for practitioners who rely on traditional development. Indeed, 78% of the practitioners who use traditional methods find that increasing awareness (N6-S in Fig. 5a) is a must have (39%) or a should have (39%). In contrast, only 56% of agile practitioners rank this as should or must have. Moreover, practitioners who use traditional development methods demand increased guidance for traceability. When looking at statement N2-M in Fig. 5a, 75% of the traditional practitioners argue that additional guidance is either a must have (31%) or a should have (44%), while for agile practitioners, their 'should have' and 'must have' sum up to 44% only.

4.3 Type of software system—RQ3

We analyze RQ3 (safety-critical vs. non-safety-critical systems) in a similar way to what we did for RQ2.

Figure 6a reveals some differences in the perception about the current practices: the costs outweigh benefits (C3-M) - 32% say this as a problem for safety critical versus 48%; ad-hoc traceability practices (C5-M)—66% for safety critical versus 48%; low stakeholder collaboration (C8-S)—(63% for safety critical vs. 45%; and unclear roles and responsibilities (C9-S): 65% safety critical versus 50%. However, when we test these differences using Mann–Whitney's U (analogously to Table 5), we do not find statistical significance: the p value is always higher than 0.05. The results are in Table 6.

When comparing the needs (Fig. 6b), the differences are even more marginal: we cannot observe any difference in the figure, nor in the statistical analysis.

Finding S9. The type of software system does not seem to have any influence on the perceived challenges, nor on the needs for traceability.

Evidence for S9. As shown earlier in this section, no statistically significant difference could be identified when splitting the current practices and the needs according to the 'Type of software system' independent variable.

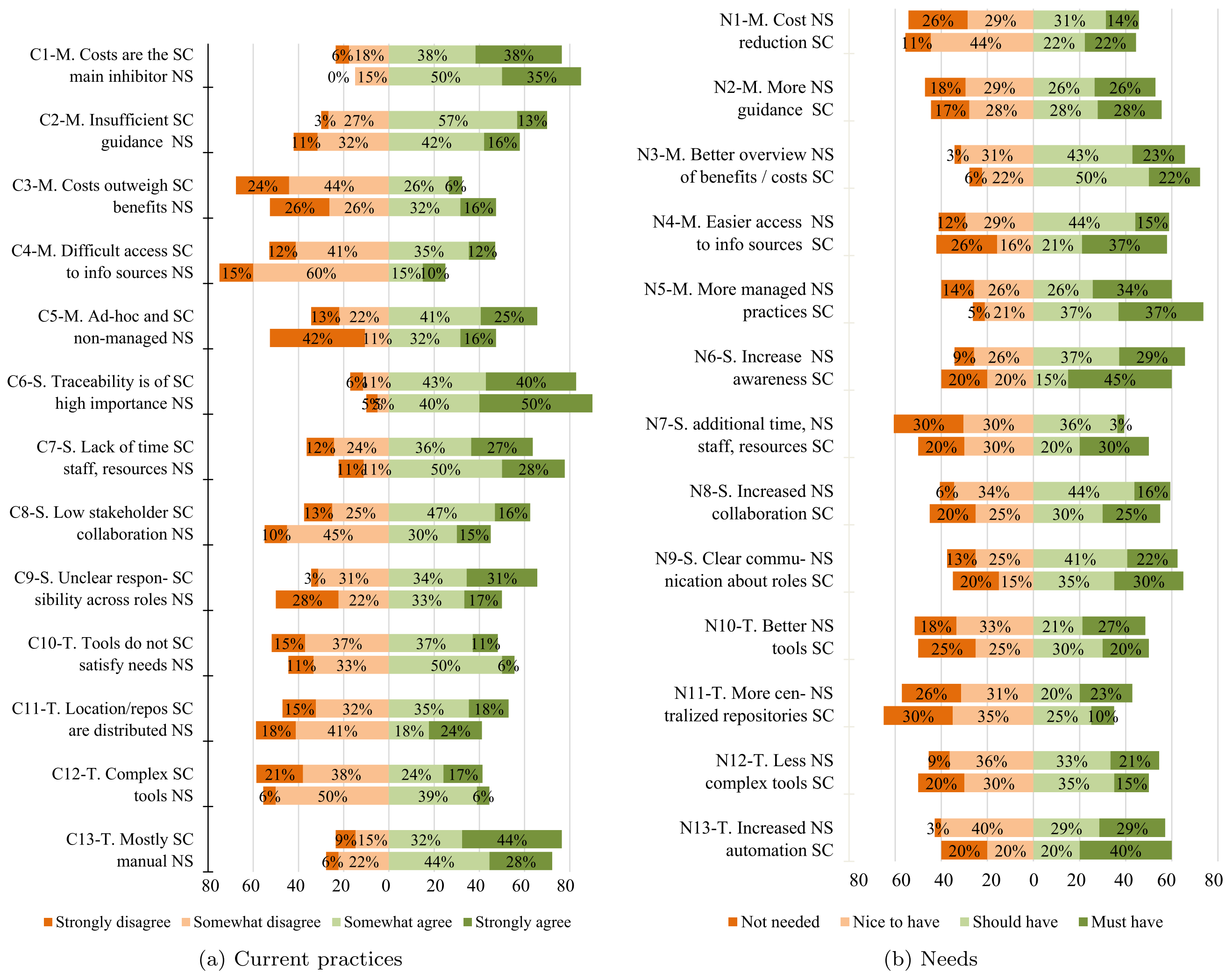

Fig. 6 Comparison of the results, splitting by the 'Type of software system' independent variable: **S**afety-**C**ritical ($n = 20$) versus **N**on-**S**afety-Critical ($n = 35$)

Table 6 Statistical significance and effect size for the safety-critical versus non-safety-critical comparison

Stat	U	z	N	*p*	*r*	Effect
C1-M	327.5	– 0.24	54	0.81	– 0.033	–
C2-M	255	– 0.671	49	0.502	– 0.096	–
C3-M	284.5	– 0.749	53	0.454	– 0.103	Small
C4-M	274	– 1.272	54	0.203	– 0.173	Small
C5-M	220	– 1.706	51	0.088	– 0.239	Small
C6-S	307	– 0.819	55	0.413	– 0.11	Small
C7-S	269.5	– 0.571	51	0.568	– 0.08	–
C8-S	281.5	– 0.766	52	0.444	– 0.106	Small
C9-S	206.5	– 1.718	50	0.086	– 0.243	Small
C10-T	231.5	– 0.284	45	0.776	– 0.042	–
C11-T	271	– 0.375	51	0.708	– 0.053	–
C12-T	245.5	– 0.359	47	0.719	– 0.052	–
C13-T	264	– 0.857	52	0.392	– 0.119	Small
N1-M	281	– 0.664	53	0.507	– 0.091	–
N2-M	298	– 0.159	52	0.873	– 0.022	–
N3-M	304	– 0.221	53	0.825	– 0.03	–
N4-M	302.5	– 0.396	53	0.692	– 0.054	–
N5-M	292	– 0.767	54	0.443	– 0.104	–
N6-S	338.5	– 0.21	55	0.834	– 0.028	–
N7-S	251	– 1.508	53	0.132	– 0.207	Small
N8-S	309	– 0.217	52	0.828	– 0.03	–
N9-S	307.5	– 0.246	52	0.806	– 0.034	–
N10-T	309.5	– 0.389	53	0.697	– 0.053	–
N11-T	308	– 0.763	55	0.446	– 0.103	Small
N12-T	292	– 0.73	53	0.466	– 0.1	Small
N13-T	346	– 0.073	55	0.942	– 0.01	–

Legend as per Table 5

5 Key findings from the interviews

We conducted interviews in order to obtain more nuanced information regarding the survey findings, by allowing the interviewees to explain the rationale behind their perception. Since the large majority of survey respondents works in Europe (Fig. 2b), we interviewed practitioners in Europe. The demographic data are presented in Table 7; the participant IDs marked with a * denote participants who did not take part in the survey, which we reached out in order to obtain a more representative sample.

We can see that the sample in Table 7 comprises a heterogeneous set of respondents. They cover a wide spectrum of work experience, from 6 months to 30 years (mean 9.73 years); various work functions (e.g., 5 software developers, 3 product owners, 2 project managers, 2 requirements analysts/engineers, etc.); organizations of different size (8 large = 200+ people, 3 medium = 50–249 people, 3 small = 10–49 people); they work in 6 different countries (5 Netherlands, 4 Germany, 2 Switzerland, 1 Norway, 1 Spain, 1 United Kingdom); they follow different development methods (3 traditional, 6 agile, 5 mixed); 4 belong to the safety-critical domain, and 6 state they are in a highly regulated domain; 6 always use traceability, 6 do it sometimes, 2 never; most employ traceability for its benefits (10), in line with the demographics for the survey of Table 7; the respondents are distributed across a variety of industry domains.

In the following, we present our findings and relate those explicitly with the survey findings by adding an identifier between square brackets (e.g., [S3,S4]) that traces back to the corresponding survey findings. The findings were derived from the topics of the questionnaire, which were as well used to tag the transcript from the interviews using the Nvivo software. Once the tagging was fully performed, we investigated the topics and search for evidence that can support the finding. Evidence is presented as quote citations from the interviewees. We looked for the evidence in the interviews to clearly present that there is an agreement on perceptions for each topic. We highlight omitted text with [...], and clarifications between parentheses: ().

Table 7 Demographics of the interviewed practitioners

ID	Work exp.	Current function	Organis. size	Country	Dev. method	Safety critical	Highly regul	Use	Reasons	Current industry
P1	20y	Project manager	Large	DE	Trad	✓	✓	Always	All (M, C, B)	Automotive
P2	9 m	Product owner	Small	NL	Agile	×	×	Sometimes	Benefits	Retail software
P3	1y	Product owner	Small	NL	Agile	×	✓	Sometimes	None	Financial
P4	6 m	Req. analyst	Large	NL	Mixed	×	✓	Always	Mandate, Benefit	Asset management
P5	5y	SW developer	Large	NL	Agile	×	×	Sometimes	Mandate	Web dev software
P6	18y	SW developer	Large	NO	Agile	×	×	Never	None	Computer software
P7	3y	SW developer	Medium	UK	Agile	×	×	Sometimes	Benefits	Video games (educational)
P8	9y	Product owner	Large	DE	Mixed	✓	×	Sometimes	Benefits	Logistics
P9*	23y	Service manager	Large	DE	Trad	×	✓	Always	Benefits	Corporate IT in satellite org
P10*	5y	Dev team lead	Large	NL	Agile	×	×	Always	Benefits	Low-code platform
P11*	6y	Req. Engineer	Medium	DE	Mixed	✓	×	Always	Benefits	Service provider
P12*	30y	Project manager	Small	CH	Trad	×	×	Never	NA	ERP systems development
P13*	10y	SW developer	Large	CH	Mixed	×	✓	Always	Benefits	Service provider
P14*	5y	SW developer	Medium	ES	Mixed	✓	✓	Sometimes	Benefits	Telecommunications

Finding I1 [S2,S5]. Tangible benefits outweighing costs are needed to observe the return of investment from implementing traceability.

Evidence for I1. In line with previous research [2], several practitioners stated that the return of investment (S2) is not evident for managers and software development teams. As argued by P5: "*Exactly, what are the expected benefits of traceability?* [...] *I want to know that this work that I am doing with traceability is going to give some benefit or improvement of quality or transparency in how everything is related to each other. The expected benefits are like minimal, or unknown right now. And costs are much more apparent.*" In this line of reasoning, practitioner P11 states: "*It is not about that much money and time.* [...] *Many people just do not see the benefits and for that reason they just do not do it.*" Even if the benefits are clear, the organization should be able to afford it. For small companies, this is not simple, as stated by P13: "*We develop cheap and small-scale software, and our clients don't even know what traceability is. They will never pay for it.*" P2 adds: "*The benefits do outweigh the costs. But as a company we are not in the position to make these costs and it costs you a lot of effort to convince people to do so.*" Moreover, those experiencing the benefits are not the ones paying for the cost; P2 states: "*The people who experience the benefits are not the people who pay the costs most of the time. So, as a product owner I am actually one of the only employees who experience both the costs and the benefits.*" Finally, the benefits depend on the individual perspective since traceability is not always enforced, as argued by P1: "*There is nothing imposed from externally that we have to do this but it is more let's say self-understanding that it makes sense, so nobody really complains about the costs.*" This supports the need for better embedding in organizational practices (S5).

Finding I2 [S4,S8.] Traceability practices and tools are not mature enough to cope with distributed software artefacts, but engineers fill the gaps.

Evidence for I2. According to the survey, access to distributed information was shown to not represent a big problem in general (S4), with agile practitioners demanding easier access (S8). Our interviewees elaborate on these aspects. In some cases, there is simply no need for easier access since most artifacts are already stored centrally or are easily accessible: P1 states "*Well, I am the expert for this component. There is no other person in the world who knows the system better than me. So, there is no need to access other people and the artifacts that I am dealing with, well, both the requirements and test cases are written by me.*" However, the need for easier access arises when there are multiple tools, as stated by P3: "*In general, information is all over the place. That was also my experience in the* *other startup I worked at. Where things come from different sources, like different people requesting stuff"*. For larger organizations, external information sources are a factor, as exemplified by P8: "*The problem is (that) all the different artifacts are related to different user rights. I have just two product owners in training that I am guiding through the process. And they started one month ago, and every day we are having the situation where they cannot do their work if they do not have the access rights.*" The problem of access rights relates to tools, as indicated by P14: "*The more centralized the software artefacts, the better. Unfortunately, in the real world, we need to trace towards external documentation. For this, our current tools cannot support us because we are lacking permits.*" Sometimes, security mechanisms introduce concerns, as stated by P13: "*Most of the time it is security problems. Some firewalls, and some strange networks problems. They are different tools, and they have to communicate.*" Conversely, P10 argues that smart people are a possible solution: "*We talked about code, merge requests, ticket support systems, stories, Jira. They are different systems, obviously, this takes effort, but we're 300 engineers here!*" These quotes allow us to conclude that there is a need for contextualizing the use cases for traceability tools. Highly distributed organizations and larger project sizes seem to have an influence on the role of the tools and the needs for providing features to support traceability among artefacts. However, further studies need to be conducted to determine with higher precision which project size and scope play an important role for defining traceability tools.

Finding I3 [S5.] No silver bullet: deciding between managed vs. ad-hoc traceability practices depends on artifacts, organizational culture, and tools.

Evidence for I3. The need for better organizational embedding (S5), in terms of guidance, is clarified by our interviewees. The bottom line is that there is no win-win situation as to the degree of management: many factors influence the decision. For small and medium organizations, management practices and organizational culture are key. For example, P5 states "*When I think of managed, then you have either this software that ensures it is being done, or you have this hierarchy, this manager making sure that it is being done. And for most of how I know of my current company, it is done per team. So pretty ad-hoc.*" P11 has a similar perception: "*Yes, because people will put these links when they need them. You know, you have this technical documentation so you put the link. And for me I try to do it always: this high-level requirement is related to this low-level requirement. But developers know they need to simply put the link to corresponding requirements.*" The type of software system matters too, as stated by P1: "*If you decide in your project to skip traceability there would not be at the first glance a manager who blames you. Maybe, if you have a highly safety-critical system, there will be a safety assessment at some point in time and then it would become obvious that you would miss traceability.*" P3 explains how the level of management is affected by development phase and artifacts: "*If you are talking about features that were planned and kinda went through this process of refinement and estimation and everything. Then I think it is easier to trace back to where it came from. Whereas, if its more like ad-hoc requirements, or things that popped up because we figured out that we needed to fix something. Then, zero traceability.*" Finally, some practitioners mentioned tools; e.g., P13 states "*Yes, well, it is managed, managers have an idea. But traceability just does not work, because there are no tools.*"

Finding I4 [S2.] Implementing traceability does not require further resources: it is about benefits, commitment, and team collaboration.

Evidence for I4. The positive cost/benefit ratio (S2) is elaborated by our interviewees. Having sufficient staff and resources does not mean good traceability practices will be implemented. That mainly depends on organizational culture and motivations. P11, for instance, stated that "*Time is not the issue, because I need to put the link anyway.* [...] *It is not how many people you have, maybe it is more: do we have to do it if we are three people, if we are 10, or 15 people, and we are located in different offices, cities or countries? So in the latter cases, it makes better sense to do it.* P3 refers to commitment as a main enabler: "*But I am also not sure that if we had more people, we would have more traceability.* [...] *I think it is a part of your product, or part of your role.* [...] *But I think for many people it is just not a priority and they just end up not doing it.*" The case of startups is exemplified by P7: "*The developers are left to almost maintain their traceability themselves, so I felt like it uses up a lot of our time.* [...] *It definitely does take up more effort and time.*" In some smaller organizations, verbal communication seems a key factor, as stated by P3: "*And also, because it is such a small company, verbal collaboration kind of replaces this traceability.*". P10 stresses the importance of teams over rigid processes: "*Team leads are responsible for educating new team members and for reminding people of what to do. Responsibility may not be explicit, but there is organic checking.*"

Finding I5 [S5.] There is no single role who is responsible for traceability.

Evidence for I5. Better embedding of traceability practices in the organizations (S5) does not imply, according to our interviewees, that a single responsible can or should be defined. This is extremely clear in startups, as stated by

P7: *'The roles are very flexible. My role is currently Unity (a game engine) developer. But I have done a bunch of other things.* [...] *It is very unclear on what our roles are.*" A similar situation applies to large companies that use agile approaches; for example, P5 stated: "*That is something for each developer, or what the project manager is more interested in it, or who should be responsible for it. Should it be everyone? Should it just be managers? Does the developer need to care, or know about the stuff or is it more the manager who cares about how the backlog and the code itself are related to each other?*" Organizational culture has a big impact too, as explained by P8: "*Maybe it is also related to our company, what I see is there is a cultural difference in general. Young people tend to plan everything,* [...] *'Country A' people, yeah, they are planning to a specific amount of degree. But at the end they are just doing. If 'Country B' people are in our setup, they are planning a lot: they are even defining what is a role.* [...] *So, there is also an intercultural aspect and within here it is the same. In 'Country C' you need to define what someone is allowed to do. If you don't define it, nobody will do it.*" P9 explains how, although no single role exists, what matters is reaching a clear agreement: "*We do not have a specific role always responsible, but there is always an agreement on who is the one who is responsible for defining the requirements: the vendor, the developer, etc. No unclarity at that time.*"

Finding I6 [S4.] A dedicated traceability tool? No, thanks: we re-purpose and customize for the user.

Evidence for I6. A major reason why tools are not found to be complex (S4) is that dedicated traceability tools may not be used that extensively: most of our respondents explained that they re-purposed their current tooling for supporting traceability.

For instance, practitioner P1 has experienced the following: "*If you ever have to use DOORS, working with traceability is okay. But of course, it could be better. If you think about drag and drop activities, on visualization and so on. Well, usually it is not that easy to see at one glance what is the overall traceability picture.* [...] *The individual traces are there, and you can jump forth and backwards.* [...] *But if you, for instance, would like to annotate something to this trace, it is nothing that is let's say, treated or considered as fun.*" Re-purposing current tools does not mean that all traceability needs are satisfied, and heavy manual work is necessary to make the re-purposed tools work properly. For example, P2 stated: "*I mean, the Atlassian suite does what it needs to do right. It allows you to create every link you could possibly dream of. The only thing is that you still need to do it manually. And I don't think they (the developers) consider that their responsibility.*" The effort required for re-purposing is also highlighted by P13: "*There should be a strong team that is taking care of all these tools. But what I have heard from managers is that 'the tools are not important, the process is important, we (managers) are important'. But, this is wrong, the tools are enabling the work, in my opinion. But they (managers) do not see it. For them (managers), tools are just toys for engineers.*" Some practitioners argue that re-purposing needs customization. For example, P11: "*If I am responsible for code, and then I put comments on BitBucket, where all is easily accessible. Those are the tools that I use. If I, as a programmer, need to find a requirement and put some traceability, then it is complicated. But if we have different tools for different people, and they use them in a way to establish traceability, then it works. Everything is customized.*"

Finding I7. Traceability automation is desired, but automation won't replace us.

Evidence for I7. The interviewees expressed a need for more traceability automation to reduce boring, repetitive tasks that require high effort, e.g., trace creation and maintenance, as exemplified by P3: "*One of the things I struggle with, using Trello, is linking stuff together. You typically have epics and user stories. Where one epic might be broken down in different user stories.* [...] *All those different things are different cards. So, I have one card for the epic, and one card for the user story, and then each user story has a number of tasks to implement the user stories. And tasks are also different cards. So, if you want to link them all together, what you need to do, is basically, literally like copy paste the Trello card URL, and go card by card, linking that.*" Although most participants would like more automation, they think any automated system would require a person to check for correctness. P1 said "*Some years ago, I had a PhD student who was working on deriving trace links between requirements and test cases automatically.* [...] *Although he made significant progress, in the end, the precision and recall values were at a level* [...] *it does not make sense to use them because of the number of false positives.*" P7 reinforced this experience and said that "*Automation is definitely nice for the small things, like attaching the ID, putting it on the board, etc. But I personally believe having a human eye to look at it as well to maybe iron out certain things that the automation may have missed.*"

Finding I8. The phase in which traceability gives the most depends on practitioners' role.

Evidence for I8. Our interviewees gave diverse opinions on the phase in the software development life-cycle in which traceability offers the highest benefits. In summary, their perception is highly aligned with their roles. P2, who is a product owner stated: "*Definitely in deployment and maintenance. Yeah that is where you find out and stuff is not working and you want to know why.* [...] *Of course, the development and implementation would be the next part, but, the biggest benefit is with the operations and the support department.*" On the other hand, P9, a service manager with clear RE responsibilities, argues that "*Design to requirements. If you start with the wrong design, the development will be wrong, etc. Also test cases to code and test cases to requirements matter. But the central aspect is design.*" People who are highly involved in development point out code-related trace links; for example, P6 says "*I think that my answers are kinda reflected that it is from code to requirements. But from bug to code and or from bug and code to pull requests/commit that is valuable.*" P5, another developer, strengthens this perspective: "*When there is a bug, we can determine if this is something that we overlooked or is this something, we made a code change, and that introduced this bug.*" and also highlighting lack of knowledge regarding earlier phases: "*Requirements gathering, yeah I guess I don't know enough of what is involved in that aspect.*"

6 Discussion

Our study provides empirical data on the barriers to software traceability in terms of current practices and needs as expressed by software engineering practitioners. The information gained from an online survey with 55 practitioners is complemented by in-depth, qualitative data from 14 practitioners.

In the following, we first present our conclusions by addressing the research questions in Sect. 6.1. Then, we discuss the threats to validity in Sect. 6.2, and we outline future directions in Sect. 6.3.

6.1 Conclusions

Before answering our research questions, we shall make the premise that various reasons exist for adopting traceability (S1 and Table 7), with the *expected benefits* being the prevalent one. Since multiple roles within the software development lifecycle have to do with traceability (I5), it is unsurprising that practitioners' perception of the phase in which traceability gives its best depends on their role (I8).

Regarding the perceived significance of the barriers to software traceability in practice (**RQ1**), we found that although costs are an inhibitors, the respondents do not find those to outweigh the benefits (S2, I4), as traceability is regarded of high importance (S3). However, better embedding in the organizations is necessary for the stakeholders to see the benefit of traceability (S5), as those benefits are hardly tangible (I1). Interestingly, our results contradict prior evidence [19, 26, 26, 39] by showing that tool complexity and difficult access to information sources do not seem to be a challenge (S4). One possible explanation is that people ("engineers" in I2) are able to fill the gaps, even when the tools are not optimal.

Regarding how the software development paradigm affects the perceived importance of barriers to software traceability (**RQ2**), the survey did not reveal striking differences between the two groups: agile and traditional. Nevertheless, some differences could be identified concerning some aspects of traceability. Team collaboration, a key trait of agile development, seems to be an enabler for implementing traceability practices (I5). Such result confirms prior evidence as reported by other authors [3, 17, 22]. Nevertheless, agile practitioners more strongly demand for easier access to those sources (S8). On the other hand, traditional practitioners stress the need of a better organizational embedding: they demand more awareness of the benefits and increased guidance (S7). The results from (S7) reinforce the evidence presented in earlier studies [5, 28, 39].

The impact of whether the software is safety critical (**RQ3**) is minimal. Our study reveals that this factor seems to affect less strongly the perceived barriers (S9). Nevertheless, further research should be conducted to confirm or disprove the lack of notable differences and evaluate the extend the evidence presented by [31] holds for these two types of software systems by characterizing low-end and high-end traceability users.

When we consider our major research goal (**MRG**) of obtaining contemporary empirical knowledge on the field, we see that traceability is of high importance for the software development lifecycle (S3), and the practitioners' perception is that the costs, although a key inhibitor for more mature practices, do not outweigh the benefits (S2). Our interviews helped us better understand these notions of benefit and cost. The practitioners pointed out that *tangible* benefits are essential (I2), especially when communicating with colleagues and managers. On the other hand, costs mostly amount to the *effort* that is necessary to establish and maintain trace links (I1); obviously, such effort needs to be balanced by the benefits. To enable this, traceability practices need a better embedding within organizations (S5), but how? According to our study, there is no silver bullet and the degree of traceability management depends on artifacts, organizational culture, and tools (I4). Also, there is *no single role* that is responsible for traceability (I6), but *individual commitment* and *team collaboration* are the key enablers (I5). Dedicated tools for traceability do not seem necessary: our interviewees clarified that their preference is to *re-purpose and customize* the tools they use for their daily tasks (I7), and that automation could represent a useful aid, but they challenge the effectiveness of current automation (I8).

6.2 Threats to validity

We analyze the threats to validity according to the categorization proposed by Wohlin et al. [38].

Conclusion validity. Random heterogeneity of respondents applies, as we reached them via an online survey. Thus, our conclusions might not reflect certain subsets of population. Also, most of our respondents are located in Europe. Furthermore, our sample size (55) requires care when splitting the responses into multiple groups, due to the limited size, especially for the minority groups. Despite the efforts of increasing the sample size by leaving the questionnaire online for a longer time and the direct invitation of circa 50 people through personal e-mails, we have to acknowledge the threat of low statistical power. To mitigate this risk, our analyses include a transparent reporting that includes divergent stacked bar charts, significance values, and effect size. The adoption of an online questionnaire may question the reliability of measures in terms of proper wording and instrument layout. To minimize this threat, we conducted pilots that tested both language formulation and content.

Internal validity. A threat with surveys is finding a representative group. This problem is limited in our case, for our target audience is rather broad and we did not mean to specify strict constraints. To avoid bias, in our advertising and personal e-mails, we tried to be inclusive by indicating we were looking for participants with an opinion or experience on traceability, without expressing a polarity (e.g., traceability enthusiast or detractor). Regarding the interviews, bias exists in the sense that only people who expressed their interest in participating through the questionnaire were invited, besides the 6 additional participants. However, this threat is minimized given the pretty diverse distribution of our interviewees (see Table 7) in terms of development method, safety critical, organization size. Another possible risk is that some of the respondents indicated that they currently do not use traceability; we decided to retain them in our sample as they indicated experience in the field and because of the perception-based formulation of our questions (Table 2).

Construct validity. By conducting a questionnaire-based survey, we tried to minimize the influence of the researchers on the participants. We opted for simple language to reduce ambiguity, and we conducted pilots that tested both language formulation and content. We would have liked to test the survey with additional practitioners, but this was not possible due to time and resource constraints. Despite these efforts, some statements probably suffer from ambiguity and could still be improved. For example, the results from findings S4, I2, and I6 might seem controversial regarding the complexity and role of tools for traceability. The participants might not have the same understanding regarding needs and use of traceability tools in different software projects. Despite the participants did not indicate problems regarding terminology in the final page of the questionnaire, we consider there is a threat regarding the contextualization since it seems it was not explicit enough for the participants. These experience should be consider for further replications and similar studies. To mitigate the influence of the researchers in the interviews, we followed the structure of the questionnaire and paid attention to letting interviewees express their opinions freely. The interview, however, allowed for additional clarifications that the survey did not support: the participants could ask clarifying questions. Therefore, we excluded the results from the 6 additional interviews in the survey results, although the participants expressed their sentiment and priority about the statements. The classification of the statements into the Managerial, Social, and Technical categories proposed by Bouillon et al. [6] is at the basis of finding S6. This finding, which shows no difference in the perceived importance across these categories, assumes that the statements within these categories are a truthful representation of each category. To minimize bias, we used the very classification introduced [6], without introducing additional statements or removing statements: each statement corresponds to one of the obstacles that that work identified. With different statements, however, the results could be different.

External validity. Our sample consisted of 55 participants. We were able to gather data from a diverse group of participants. Nonetheless, we do see that most of that participants are from Europe, work for large companies, and 2/3 work with agile methods. Therefore, we need to be cautious on the generalizability of the results. Regarding the interviews, although we succeeded in attracting different roles, we shall note that all of them work in Europe. On the one hand, the opinions are expected to reflect the results of the majority of the population that participated in the questionnaire (82% of the participants were from Europe). On the other hand, we acknowledge that another sample involving participants from different geographical areas may lead to different results. Similarly, the interviewees we approached may not exactly represent the 55 participants (or the entire population) when it comes to their experience with traceability *tools*: dedicated studies may be necessary regarding this specific aspect.

6.3 Future work

Our study contributes to updating and strengthening the body of knowledge regarding software traceability in practice. We hope our results pave the way for additional studies and can be employed to direct the research efforts toward the actual practices and needs of the practitioners.

Our questionnaire, which is available in the online appendix (https://zenodo.org/records/8021723), can be re-used for additional studies to increase the sample size or to focus on other regions of the world: our results mostly hold for Europe. Since practitioners do not seem eager to change their working tools, it is urgent to explore how existing solutions could be effectively embedded within the tools that are used in practice, e.g., Jira. Also, there is still space to investigate how automation may help human analysts and other roles in software development, like some studies started exploring [29]. An important factor to study is collaboration; existing studies such as Wohlrab et al.'s [39] need to be conducted in diverse development contexts to shed light on which practices may be adopted to effectively enable traceability.

Funding Open access funding provided by ZHAW Zurich University of Applied Sciences.

References

1. Antoniol G, Cleland-Huang J, Hayes JH, Vierhauser M (2017) Grand challenges of traceability: the next ten years. arXiv:1710.03129
2. Arkley P, Riddle S (2005) Overcoming the traceability benefit problem. In: Proceedings of the IEEE international requirements engineering conference (RE), pp 385–389
3. Batot ER, Gérard S, Cabot J (2022) A survey-driven feature model for software traceability approaches. In: Johnsen EB, Wimmer M (eds) Fundamental approaches to software engineering. Springer, Cham, pp 23–48
4. Beck K, Beedle M, Van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, et al (2001) Manifesto for agile software development
5. Blaauboer F, Sikkel K, Aydin MN (2007) Deciding to adopt requirements traceability in practice. In: Proceedings of the international conference on advanced information systems engineering (CAiSE), pp. 294–308
6. Bouillon E, Mäder P, Philippow I (2013) A survey on usage scenarios for requirements traceability in practice. In: Proceedings of the international working conference on requirements engineering: foundation for software quality (REFSQ), pp 158–173
7. Cleland-Huang J (2006) Just enough requirements traceability. In: Proceedings of the international computer software and applications conference (COMPSAC), pp 41–42. https://doi.org/10.1109/COMPSAC.2006.57
8. Cleland-Huang J (2012) Traceability in agile projects. In: Software and systems traceability. Springer, pp 265–275
9. Cleland-Huang J, Berenbach B, Clark S, Settimi R, Romanova E (2007) Best practices for automated traceability. Computer 40(6):27–35
10. Cleland-Huang J, Gotel OOC, Huffman Hayes J, Mäder P, Zisman A, Hayes JH, Mäder P, Keyes M, Zisman A (2014) Software traceability: trends and future directions. In: Proceedings of the session on the future of software engineering (FOSE), pp 55–69
11. Cohen J (2013) Statistical power analysis for the behavioral sciences. Academic Press
12. De Lucia A, Marcus A, Oliveto R, Poshyvanyk D (2012) Information retrieval methods for automated traceability recovery. In: Software and systems traceability. Springer, pp 71–98
13. Forsberg K, Mooz H (1992) The relationship of systems engineering to the project cycle. Eng Manag J 4(3):36–43
14. Fritz CO, Morris PE, Richler JJ (2012) Effect size estimates: current use, calculations, and interpretation. J Exp Psychol Gen 141(1):2
15. Furtado F, Zisman A (2016) Trace++: a traceability approach to support transitioning to agile software engineering. In: IEEE international requirements engineering conference (RE), pp 66–75
16. Gotel O, Cleland-Huang J, Hayes JH, Zisman A, Egyed A, Grunbacher P, Antoniol G (2012) The quest for ubiquity: a roadmap for software and systems traceability research. In: Proceedings of the IEEE international requirements engineering conference (RE), pp 71–80
17. Gotel O, Cleland-Huang J, Hayes JH, Zisman A, Egyed A, Grünbacher P, Dekhtyar A, Antoniol G, Maletic J (2012) The grand challenge of traceability (v1.0). In: Software and systems traceability, pp 343–409
18. Gotel O, Finkelstein A (1994) An analysis of the requirements traceability problem. In: Proceedings of the IEEE international conference on requirements engineering (RE), pp 94–101
19. Gotel O, Finkelstein C (1994) An analysis of the requirements traceability problem. In: Proceedings of IEEE international conference on requirements engineering, pp 94–101. https://doi.org/10.1109/ICRE.1994.292398
20. Guo J, Cheng J, Cleland-Huang J (2017) Semantically enhanced software traceability using deep learning techniques. In: Proceedings of the IEEE/ACM international conference on software engineering (ICSE), pp 3–14. https://doi.org/10.1109/ICSE.2017.9
21. Guo J, Monaikul N, Cleland-Huang J (2015) Trace links explained: an automated approach for generating rationales. In: IEEE international requirements engineering conference (RE), pp 202–207
22. Hernández R, Moros B, Nicolás J (2023) Requirements management in DevOps environments: a multivocal mapping study. Requir Eng. https://doi.org/10.1007/s00766-023-00396-w
23. Ingram C, Riddle S (2013) Cost-benefits of traceability. In: Software and systems traceability, pp 23–42
24. Kuang H, Nie J, Hu H, Rempel P, Lü J, Egyed A, Mäder P (2017) Analyzing closeness of code dependencies for improving IR-based traceability recovery. In: IEEE international conference on software analysis, evolution and reengineering (SANER), pp 68–78
25. Mäder P, Egyed A (2015) Do developers benefit from requirements traceability when evolving and maintaining a software system? Empir Softw Eng 20(2):413–441. https://doi.org/10.1007/s10664-014-9314-z
26. Mäder P, Gotel O, Philippow I (2009) Motivation matters in the traceability trenches. In: Proceedings of the IEEE international requirements engineering conference (RE) pp 143–148. https://doi.org/10.1109/RE.2009.23
27. Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. Ann Math Stat 18:50–60
28. Maro S, Steghöfer JP, Bozzelli P, Muccini H (2022) Tracimo: a traceability introduction methodology and its evaluation in an agile development team. Requir Eng 27(1):53–81. https://doi.org/10.1007/s00766-021-00361-5

29. Maro S, Steghöfer JP, Hayes J, Cleland-Huang J, Staron M (2018) Vetting automatically generated trace links: what information is useful to human analysts? In: Proceedings of the IEEE international requirements engineering conference (RE), pp 52–63
30. Rahimi M, Cleland-Huang J (2018) Evolving software trace links between requirements and source code. Empir Softw Eng 23(4):2198–2231
31. Ramesh B (1998) Factors influencing requirements traceability practice. Commun ACM 41(12):37–44. https://doi.org/10.1145/290133.290147
32. Rath M, Rendall J, Guo JL, Cleland-Huang J, Mäder P (2018) Traceability in the wild: automatically augmenting incomplete trace links. In: Proceedings of the IEEE/ACM international conference on software engineering (ICSE), pp 834–845
33. Regan G, McCaffery F, McDaid K, Flood D (2012) The barriers to traceability and their potential solutions: Towards a reference framework. In: Proceedings of the EUROMICRO conference on software engineering and advanced applications (SEAA), pp 319–322. https://doi.org/10.1109/SEAA.2012.80
34. Robson C (2002) Real world research—a resource for social scientists and practitioner-researchers, 2nd edn. Blackwell Publishing, Malden
35. Ruiz M, Hu J, Dalpiaz F (2023) Online appendix of “Why Don’t We Trace? A Study on the Barriers to Software Traceability in Practice”. https://doi.org/10.5281/zenodo.8021723
36. Sommerville I (2015) Software engineering, 10th edn. Addison-Wesley
37. Wang W, Niu N, Liu H, Niu Z (2018) Enhancing automated requirements traceability by resolving polysemy. In: Proceedings of the of the IEEE international requirements engineering conference (RE), pp 40–51
38. Wohlin C, Runeson P, Hst M, Ohlsson MC, Regnell B, Wessln A (2012) Experimentation in software engineering. Springer
39. Wohlrab R, Knauss E, Steghöfer JP, Maro S, Anjorin A, Pelliccione P (2018) Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture. Requir Eng. https://doi.org/10.1007/s00766-018-0306-1