Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture

Citation for the original published paper (version of record):
Wohlrab, R., Knauss, E., Steghöfer, J. et al (2020). Collaborative traceability management: a multiple case study from the perspectives of
organization, process, and culture. Requirements Engineering, 25(1): 21-45.
http://dx.doi.org/10.1007/s00766-018-0306-1

Requirements Engineering (2020) 25:21–45
https://doi.org/10.1007/s00766-018-0306-1

ORIGINAL ARTICLE

Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture

Rebekka Wohlrab[1,2,3] · Eric Knauss[1] · Jan-Philipp Steghöfer[1] · Salome Maro[1] · Anthony Anjorin[4] · Patrizio Pelliccione[1,5]

Received: 4 May 2018 / Accepted: 8 November 2018 / Published online: 21 November 2018

Abstract

Traceability is crucial for many activities in software and systems engineering including monitoring the development progress, and proving compliance with standards. In practice, the use and maintenance of trace links are challenging as artifacts undergo constant change, and development takes place in distributed scenarios with multiple collaborating stakeholders. Although traceability management in general has been addressed in previous studies, there is a need for empirical insights into the *collaborative* aspects of traceability management and how it is situated in existing development contexts. The study reported in this paper aims to close this gap by investigating the relation of collaboration and traceability management, based on an understanding of characteristics of the development effort. In our multiple exploratory case study, we conducted semi-structured interviews with 24 individuals from 15 industrial projects. We explored which challenges arise, how traceability management can support collaboration, how collaboration relates to traceability management approaches, and what characteristics of the development effort influence traceability management and collaboration. We found that practitioners struggle with the following challenges: (1) collaboration across team and tool boundaries, (2) conveying the benefits of traceability, and (3) traceability maintenance. If these challenges are addressed, we found that traceability can facilitate communication and knowledge management in distributed contexts. Moreover, there exist multiple approaches to traceability management with diverse collaboration approaches, i.e., requirements-centered, developer-driven, and mixed approaches. While traceability can be leveraged in software development with both agile and plan-driven paradigms, a certain level of rigor is needed to realize its benefits and overcome challenges. To support practitioners, we provide principles of collaborative traceability management. The main contribution of this paper is empirical evidence of how culture, processes, and organization impact traceability management and collaboration, and principles to support practitioners with collaborative traceability management. We show that collaboration and traceability management have the potential to be mutually beneficial—when investing in one, also the other one is positively affected.

Keywords Traceability management · Collaboration · Software development processes · Multiple case study

✉ Rebekka Wohlrab
wohlrab@chalmers.se

Eric Knauss
eric.knauss@cse.gu.se

Jan-Philipp Steghöfer
jan-philipp.steghofer@cse.gu.se

Salome Maro
salome.maro@cse.gu.se

Anthony Anjorin
anthony.anjorin@upb.de

Patrizio Pelliccione
patrizio.pelliccione@cse.gu.se

[1] Chalmers University of Technology, Gothenburg, Sweden

[2] University of Gothenburg, Gothenburg, Sweden

[3] Systemite AB, Gothenburg, Sweden

[4] Paderborn University, Paderborn, Germany

[5] University of L'Aquila, L'Aquila, Italy

1 Introduction

Traceability helps practitioners manage increasingly complex software projects in industry [19, 55], e.g., by supporting change impact analyses and monitoring connections between artifacts [20]. As a consequence, traceability positively impacts the development speed [29].

In this paper, we adopt Gotel et al.'s understanding of traceability [20]: Software *traceability* is the ability to create and use links between artifacts. This allows, for instance, users of traceability to connect a requirement to its origin and to any other artifacts used in subsequent phases in the software lifecycle (e.g., its specification or affected lines of source code). These connections are called *trace links* and connect a *source artifact* to a *target artifact*. Different *types* of *artifacts* can be distinguished, e.g., a requirement, an element in a model, a line of code, an issue in a bug tracking system, or the result of a test case. We define *traceability management* as the planning, organization, and coordination of all tasks concerned with traceability. This includes, for example, the creation, maintenance, and use of trace links.

Real-world industrial projects are characterized by stakeholders with different backgrounds, working in distributed teams, at multiple locations [54]. One of the open challenges formulated by the traceability research community has been to achieve full traceability in large-scale development contexts, where multiple stakeholders are involved and collaborate across organizational borders [20]. At the same time, interest in collaborative software engineering and collaborative modeling is unabated (see, e.g., [15], which also mentions that traceability is necessary in collaborative model-driven software engineering), but has not been discussed in the context of traceability yet. Our previous work [58] has focused on *collaborative* aspects of traceability management. We identified challenges that are related to the fact that multiple stakeholders need to work together to achieve traceability goals, often across tool and organizational boundaries. Apart from challenging factors, we also found that collaboration can be positively affected by traceability management and vice versa. This study extends the previous work toward a theory of collaborative traceability management. We investigate two additional research questions, create practical principles to support systematic collaborative traceability management, and provide a definition of collaborative traceability management as follows:

Collaborative traceability management is the collaborative planning, organization, and coordination of all tasks concerned with traceability in multi-person projects across organizational, discipline, or tool boundaries.

Building on our previous work, we saw the importance to shed light on how collaborative aspects relate to how traceability management is conducted in practical development environments. In this paper, we extend our previous findings on challenges of collaborative traceability management and effects of traceability management on collaboration [58], as phrased in the following research questions:

- **RQ1:** What are practitioners' challenges with collaboration in traceability management?
- **RQ2:** How can traceability management support collaboration?

We found that RQ1 and RQ2, as the topic of collaborative traceability as a whole, need to be understood in the context of the used industrial traceability management approaches. Approaches to traceability management have been addressed by related work (e.g., [8, 47]), but never in relation to collaborative aspects:

- **RQ3:** How does collaboration relate to different approaches of traceability management?

Traceability is always integrated in and characterized by the context of the development effort, characterized by culture, organization, and processes. There exists related work focusing on processes and underlying paradigms (e.g., agile traceability management) [7, 13, 16], however, not focusing on how collaborative aspects of traceability management are influenced by characteristics of the development effort. For this reason, we specified our final research question as follows:

- **RQ4:** What characteristics of the development effort influence traceability management and collaboration?

This paper extends our previous contribution [58] significantly: We investigate additional research questions (RQ3 about current traceability approaches, and RQ4 about the influence of characteristics of the development effort on traceability management), discuss them in detail, and establish connections to research questions RQ1 and RQ2. We believe that providing additional information about traceability management and collaboration approaches and characteristics of the development effort also improves the reproducibility of the study. Finally, we extend the discussion of our previous findings focused on how our findings are compared to existing literature. Based on this additional material, also reflected in our discussion of related work, we draw conclusions about the implications of our findings for practitioners and researchers. Concretely, we create principles that support practitioners to conduct more systematic collaborative traceability management. We believe that this additional context increases the value of our findings toward a theory of collaborative traceability management.

We answered these research questions using an exploratory multiple case study. We conducted semi-structured

interviews with 24 practitioners from 15 industrial cases. Our study presents empirical evidence that is independent of any concrete tool solutions. We thus extend the current body of knowledge on traceability management with the following contributions:

- Challenges with collaboration in traceability management based on **RQ1**.
- The positive effect of traceability on collaboration based on **RQ2**.
- An analysis of current traceability management approaches and their relation to collaboration, based on **RQ3**.
- Characteristics of the development effort that influence traceability management and collaboration, identified by investigating **RQ4**.
- Practical principles for collaborative traceability management based on a discussion of our findings.

Based on this collaborative perspective on traceability, we suggest that future research focuses on the enabling potential of traceability. Our principles are a starting point for practitioners and researchers to create more concrete practices. In particular, we recommend the development of tool features to share and discuss information about trace links, considering the issue of information overload so that users are not overwhelmed by the amount of information presented.

This paper is structured as follows: Sect. 2 presents related work. In Sect. 3, we describe our research methodology. Section 4 describes challenges (RQ1), and Sect. 5 the positive effects of traceability management on collaboration (RQ2). In Sect. 6, we elaborate on collaboration and current traceability approaches (RQ3) and present the influence of development characteristics on traceability management and collaboration (RQ4) in Sect. 7. In Sect. 8, we discuss our findings with respect to related work and motivate our principles for collaborative traceability management. The principles are summarized together with implications for research and practice in Sect. 9. We conclude this paper and discuss future work in Sect. 10.

2 Related work

In this section, we discuss three categories of related work: (1) empirical studies on general traceability approaches, (2) traceability in agile contexts, and (3) collaborative traceability management.

2.1 Empirical studies on traceability approaches

While a number of empirical studies have been published on traceability, few studies are aimed at describing the state of the practice of traceability and practical challenges in general. Most studies focus on validating specific technical approaches [1, 28, 53], or specific aspects of traceability such as assessment [43] and benefits of traceability [29].

Klimpke and Hildenbrand [25] conducted five interviews to identify how practitioners use traceability. The majority of the companies were "high-end users" of traceability, aiming to support verification and to track project progress. In all five companies, traceability starts from the requirements level. In contrast, our study includes a significantly larger sample as well as a wider spectrum of roles. We found that while such requirements-centered approaches are common in practice, there also exist traceability approaches focusing on implementation artifacts such as issues and code changes.

Mäder et al. [30] identified four categories of traceability users: (1) the regulated group, where traceability is mandated by standards, (2) the sub-contractor group, which inherits traceability requirements passed from the companies they supply to, (3) the consultants, who design traceability strategies for their clients, and (4) the enthusiasts, who practice traceability not because it is mandatory but because they see its benefit. They conclude that there is a need for more guidance on traceability, better tools, and more empirical studies on how traceability is established and used in practice, especially across boundaries.

With respect to traceability goals, Bouillon et al. [6] report traceability usage scenarios and conclude that traceability is mainly used for tracking the origin and rationale of requirements, for tracking project progress, and for certification. The authors also report on the use of trace links for collaboration: Trace links help to identify relevant stakeholders and notify them about changes.

Gotel et al. [18] collected data from more than 100 practitioners focusing on the requirements traceability problem. Besides reporting on challenges, we also investigate detailed practices and their situatedness in development processes found within the companies. Ramesh [39] investigates how organizational, environmental, and technical factors influence requirements traceability practices. In his study, he distinguishes between high-end users and low-end users of traceability and gives a list of factors that affect traceability and recommendations for how companies can transition from low-end to high-end traceability practices. While some traceability practices are reported, the rise of the agile paradigm calls for a new study of these aspects.

Arkley and Riddle [2] investigate traceability challenges and conclude that it is essential to select appropriate roles for traceability management. Due to their knowledge of the system, developers are better suited to create high-quality links than, for example, separate quality assurance teams. However, while developers could create high-quality links, they often do not see the direct benefits of traceability. The authors suggest that organizations should ensure that

goals and needs of developers are addressed in traceability development contracts that define traceability management of shared artifacts. The findings of our study suggest that the potential of using traceability for collaboration can in fact motivate developers to see a benefit of investing in traceability.

2.2 Traceability in agile

As pointed out by Cleland-Huang [7], traceability in projects following agile methodologies is just as important as in non-agile ones. The goals are essentially the same, but the available artifacts are different and some of the documentation that is enforced in non-agile projects to, e.g., prevent embellishment, is omitted in favor of trust within the team and direct communication. Trace links commonly exist between requirements and test cases and can be accompanied by references to code in commit messages and between requirements. Cleland-Huang also suggests a classification of agile projects [7] in (i) small-to-medium-sized agile projects, (ii) large-scale, distributed, or long-lived projects, and (iii) safety-critical projects. These classes differ in their traceability needs and in the applicable techniques, e.g., relying only on direct communication is not feasible in large-scale distributed projects. In this paper, we not only investigate the impact of process on traceability, but also conduct a broader investigation on characteristics of the development process on collaboration and traceability management (RQ4).

Espinoza and Garbajosa expand on the idea of specific traceability concerns in agile projects [13] and argue that the lack of formal documentation and formal requirements calls for traceability practices that go beyond those of non-agile projects. Accordingly, the key is to closely connect requirements and tests via trace links. To do this, they propose a traceability information model, concrete roles involved in managing traceability, and rules that determine if and when links have to be established. According to our findings, even companies following more rigorous agile methodologies are not yet embracing such formally defined traceability approaches. However, the suggestions made by Espinoza and Garbajosa could be the basis for adopting traceability practices in such cases.

Gayer et al. [17] give a concrete example of integrating traceability in an agile context. In their example, trace links between different artifacts are established in a rigorous manner (e.g., user stories, test cases, and architecture models). The authors argue that the time required to create trace links was offset by the benefits, including easier sprint planning as well as "a structured way to communicate requirements among stakeholder groups." This corresponds to our findings that traceability can indeed be an enabler for collaboration.

Furtado and Zisman [16] focus on how traceability can help organizations transition to the agile paradigm. In particular, they argue that traceability can help to mitigate the typical lack of understanding of a project's "high-level scope" or issues with control by management. Trace links between artifacts created following the plan-driven paradigm and artifacts created following the agile paradigm can help to address these challenges. Our findings suggest additional traceability management-related differences and similarities between agile and "traditional" projects.

2.3 Collaborative traceability management

The topic of traceability management can be targeted with different levels of human involvement. For instance, trace links can be automatically recovered, recommended by a classifier [42], or evolved over time using tool support [38]. However, it was found to be beneficial to also involve human decision making in trace link evolution [38]. An underlying issue is that practitioners are often hesitant because of the quality of automatically generated trace links—and "prefer no links at all to (possibly) inconsistent links" [31]. We found the importance of seeing traceability management as a collaborative effort of stakeholders who leverage it to support collaboration and create incentives for individuals to create, maintain, and use trace links together.

Demuth et al. [11] conducted a study on how to use traceability for systems engineering to facilitate change notification and consistency checking of artifacts. They describe how traceability can facilitate communication between departments enabling them to identify what artifacts and stakeholders will be affected by a change. According to the authors, the main challenge in providing traceability in such a scenario is the diversity of tools that complicates traceability between artifacts stored in different tools.

With respect to tooling, Figueiredo et al. [14] describe a tool that facilitates collaboration in a distributed setting by identifying developers affected by a change via trace links. Helming et al. [21] present a tool that also uses trace links to notify users of changes; this technique is able to generate a lower number of relevant notifications compared to standard notification techniques by leveraging modifier, creator, and assign information. Strašunskas [56] discusses traceability in collaborative development environments with a strong focus on technical aspects such as how to version and store artifacts (including trace links). While Strašunskas focuses on how such artifacts can be edited in a collaborative environment, our focus is on traceability practices and challenges in such environments. Cleland-Huang et al. [9] identify a need for further empirical studies that shed light on aspects of social collaboration tools and their impact on traceability management. This is related to our investigation of how collaboration is affected by traceability management.

Table 1 Participating projects and interviewees in the case study

Case	Country	Domain	Nr. of project members	Project duration	Distribution of sites	Interviewees
1	SE	Embedded	10–30	3–4 years	One local site	Development Manager (3 years) Project Manager (13 years)
2	SE	Electrical equipment	~ 20	> 8 years of development	Multiple int. sites	Developer (10 years) Team Leader Development (> 10 years) Project Manager (> 2 years)
3	DE	Automotive	> 20	> 2 years	Multiple int. sites	Team Leader Subgroup (3 years)
4	DE	Automotive	~ 30	4 years	Multiple nat. sites	Quality Analyst (1 year)
5	SE	Telecommunications	6–8	1.5 years	One local site	Project Manager (> 2 years) System Manager (12 years)
6	SE	Automotive	15	2 years	Multiple nat. sites	Developer (3 years)
7	SE	Automotive	> 100	3 years	Multiple nat. sites	Software Architect (3 years)
8	SE	Software development	~ 4	< 6 months	One local site	Application Engineer (1.5 years) Chief Technical Officer (5 years)
9	SE	Industrial automation	25	< 1 year	One local site	Software Quality Manager (> 1 year) Head of R&D (3 years)
10	DE	Automotive	10–20	1–2 years	Multiple int. sites	Product Manager (14 years)
11	DE	IT services	50	5 years of development	Multiple int. sites	Head of Development (9 years) Developer (8 years)
12	DE	Banking self-service automation	10–50	6 months	Multiple int. sites	Head of Software Quality Assurance (6 years) Head of Project Mgmt. Office (2.5 years)
13	SE	Telecommunications	1–4	< 1 year	Multiple int. sites	Team Leader (2 years)
14	DE	Automotive	≥ 100	2–3 years	Multiple int. sites	System Software Architect (8 years)
15	DE	Software development	20–40	6 months	Multiple int. sites	Quality Manager (> 1 year) Head of Test Management (5 years)

3 Research methodology

Given that our research questions are of an *exploratory* nature and we did not know all relevant variables of the phenomenon before the study, we decided to conduct an *exploratory multiple case study* [12]. We opted for a *multiple* case study to investigate the phenomenon from several angles. *Data (source) triangulation*, i.e., considering several sources, occasions, and individuals, helped us to mitigate common threats to validity, as discussed in Sect. 3.1.

Selection of participants As our study is widely exploratory, we relied on a multi-stage purposeful sampling strategy [36]. As the research objective came up in a research project, we started with emergent sampling in this environment. In parallel with conducting interviews, we then employed a maximum variation strategy to arrive at a broad sample. We considered different project sizes and durations, different domains, and different development paradigms. For example, we made sure that we had plan-driven and agile paradigms in the mix. We concluded with snowballing [4], asking interviewees to give us names of other potential subjects, to find more similar cases within the variations. We asked the companies for interviews with individuals with different roles and at least one year of experience in the current position. While this approach can lead to a variation in the number of interviewees per company, it gives us a broad and deep sampling of cases.

In total, 24 interviewees from 15 projects agreed to participate in our study. All projects belong to different companies, except for Case 5 and 13. Table 1 shows the selected cases, their countries (Sweden (SE) or Germany (DE)), their domains, characteristics of the project, distribution of sites, and the roles and experience of the interviewees.

Data collection We collected qualitative data using *semi-structured interviews* [35]. This form of data collection allowed us to guide the interviews' flow, but allow a natural

development of the conversation. We prepared the interviews by creating an interview guide[1] with both open-ended and specific questions. The interview guide was reviewed, tested, and refined using two pilot interviews. During the interviews, we used mirroring techniques [35], i.e., we paraphrased and used the interviewees' answers to pose subsequent questions.

We reserved a period of 11 weeks between October 2015 and January 2016 for interviews, and the duration of interviews was between 45 and 120 min, with an average length of approximately 75 min. We started the interviews by asking for permission to record them. We obtained permission to record all interviews, and transcription of these recordings yielded 475 pages of transcribed text. A majority of the interviews were held in the interviewees' native tongues.

Data analysis For the data analysis, we followed Cresswell's approach [10] and adjusted it to our needs.[2] The data analysis involved studying and *coding* the collected data. Based on our main research questions, we identified central topics for this research and created a priori codes. We coded the interviews by reading the transcripts line by line and finding appropriate codes for statements. We used an *editing approach* [45], starting with the "a priori" codes and creating new codes where new interesting topics came up. We continuously revised, merged, and split the codes.

In order to ensure consistency with the use and meanings of the codes, the data analysis was initially conducted by one of the researchers. As a next step, we conducted a coding workshop in a group to discuss the codes and their relationships. The final set of codes consists of approximately 30–40% a priori codes. However, many initial codes evolved, and sub-codes were created. For example, the a priori code "Collaboration" was revised into ten sub-codes, in order to cover different collaborative aspects that were of interest to our interviewees.

These codes were grouped into candidate themes in our coding workshop and further explored by going back to the interview transcripts where needed. In several iterations, we refined the themes, created visuals, and documented our findings. Besides the interviewer and three senior researchers, the participants of this workshop included a representative of the interviewees who contributed with his industrial experience and helped to discuss implications for practitioners. We used *cross-case synthesis* [46, p. 69] to identify differences between interview results from different cases.

Reporting and quality assurance of research findings During data analysis and reporting, we maintained a clear *chain of evidence* [45] to document our procedure and results, ensuring that they remain comprehensible for the reader. To reassess our research findings, we used a *member checking* technique to get feedback from our interviewees. We contacted them with our reported research findings and asked them for their ideas and reflections. Member checking helps participants reflect on the findings and find ways to adapt their practices [50]. Our participants confirmed the research findings.

3.1 Research validity

We discuss six categories of threats to validity in qualitative research presented by Maxwell [33, 34]:

Descriptive validity Although one cannot completely avoid this threat, member checking [50] is one way to mitigate descriptive validity. Moreover, it was helpful to create transcripts to allow for a word-by-word analysis of the interviews. Aspects such as gestures, pauses, or irony were, however, not captured in the transcripts which might have resulted in misinterpretation of the statements. In cases of doubt, we therefore referred to the recordings to clarify confounding parts in the transcripts.

Interpretative validity At the beginning of the interviews, we ensured that there was a common understanding of the terminology of traceability. Most interviews were conducted in the native language of the interviewees or with interviewees with an advanced level of English. While this helped prevent misunderstandings, we faced another difficulty: Selected quotes were translated from German to English. We tried to maintain the meaning of the translated statements in the translations and discussed critical parts with other researchers to mitigate the threat of translation inaccuracies.

Theoretical validity One threat of validity is the potential of interpreting data from a biased perspective because of preconceptions we built up during this study. We potentially interpreted data differently that were not in line with our preconceptions. Feedback from fellow researchers helped to mitigate this threat. We discussed our methods and findings throughout the study and especially during the analysis.

Researcher bias When analyzing qualitative data, there is a threat of being biased by the researchers' own background, values, and theories. However, in our study, we did not have any incentives to arrive at certain conclusions, which would have been different if we had evaluated a method or tool. Collecting feedback from fellow researchers helped to

[1] https://www.dropbox.com/s/id51h5hp2g6bpez/interview_guide.pdf?dl=0.

[2] For more details on our analysis method, see https://www.dropbox.com/s/rxzkaqflttrtr0y/analysis_guide.pdf?dl=0.

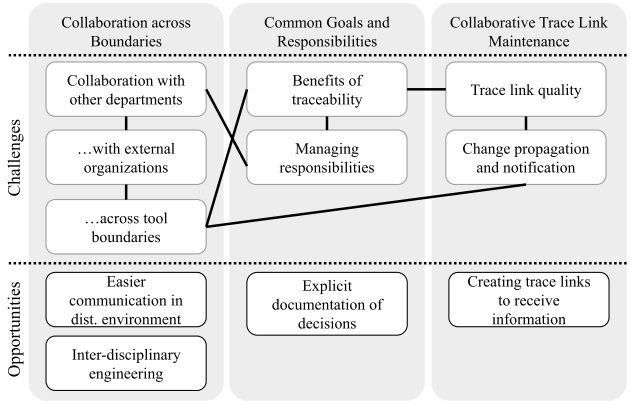

Fig. 1 Challenges and opportunities with collaborative traceability management

mitigate this threat, as well as using member checking to collect feedback from interviewees.

Reactivity: A researcher always influences the setting of an interview [34]. The way questions are phrased and how one reacts to answers influence subsequent answers in the interviews. Formulating the questions as unbiased as possible and validating them in pilot interviews was one strategy to mitigate this. We could also identify misleading questions and find appropriate ways to react to (surprising) answers during the pilot interviews.

Generalizability: Generalizability is concerned with the transferability of our findings to other situations or cases than the ones we focused on in this case study [33]. The objective of qualitative research is to describe the particular cases rather than arriving at generalizable conclusions [10].

We were able to study the phenomenon traceability from different angles using *triangulation* [46]. There might, however, be a sampling bias as we asked for participants who were knowledgeable in the area of traceability and probably had a positive attitude toward the topic.

4 Challenges: managing traceability collaboratively

In this section, we present challenges that interviewees reported and answer *RQ1: What are practitioners' challenges with collaboration in traceability management?* Figure 1 shows an overview of our findings regarding RQ1 and RQ2. In this section, we present the challenges regarding (1) collaboration across boundaries (Sect. 4.1), (2) common goals and responsibilities (Sect. 4.2), and (3) collaborative trace link maintenance (Sect. 4.3).

4.1 Collaboration across boundaries

The necessity of coordinating traceability management activities across disciplinary and tool boundaries greatly complicates traceability management.

4.1.1 Collaboration with other departments

Most of our case companies are organized in separate departments, often grouped according to discipline. A discipline is an area of expertise on a specific topic related to engineering, including system, software, electrical and mechanical engineering, product management, and testing. Strategies for managing traceability are often limited to single disciplines. Case 14 was the only case with an end-to-end concept for traceability, allowing stakeholders to trace pre-requirements to all phases of development and testing. In general, our case companies struggle with establishing common practices in the complete organization. Our interviewees stated that they are commonly established in a more bottom-up fashion:

> If we are talking about full traceability, from a requirement to the details in the code and the test case result, then this affects several phases of the development process and several organizational units. [...] It requires a lot of power and energy, and good management, to set something like this up in a top-down fashion. And on the other hand, there are many smaller teams where approaches come up in a bottom-up way. [...] Of course, it can happen that something comes up here and something else there, and then these two are competing solutions...
>
> (a product manager from Case 10)

Various interviewees stated that this is problematic in their companies. In Case 3, for example, it is possible to establish trace links, but their traceability practices are inconsistent:

> You can create these links already now, but [our current approach] is not consistent throughout the whole platform. Because I might draw links in one department but in another department someone might not know what belongs to what...
>
> (a team leader from Case 3)

Complicating factors are more likely to show in global development settings and include different backgrounds, technical terms, and tools.

> That's why we have problems now, because the communication isn't so easy. When you have time differences and language and all these barriers to communicate.
>
> (a team leader from Case 2)

Especially, our interviewees from the automotive industry struggled with this challenge. In this domain, mechanical, electrical, and software engineers develop one product together and have to find a common terminology for communication. A developer from Case 6 stresses that because of these reasons, it is *"harder to communicate efficiently and make all the developers work together."*

4.1.2 Collaboration with external organizations

Besides tracing to artifacts from other departments, it is also challenging to coordinate traceability across company borders. In general, our case companies communicate with external organizations via e-mail and manually enter relevant data into respective tools. Our cases have very little support for traceability to external organizations. Two examples are Case 2 and Case 8 where trace links exist that point to bugs or issues reported by customers. In Case 14, customers report requirements which are then imported into their requirement management tool and can be pointed to by trace links. However, in case the customers' requirements are updated, a new export and import are required. This can potentially result in inconsistencies as the trace links are not updated automatically.

Although tool boundaries are certainly a contributing factor, our findings indicate that collaboration with external organizations is still challenging even when all companies used the same tools. As the Head of Development from Case 11 points out, one of the difficulties in this context is that suppliers collaborate with several customers and are not willing to adapt their traceability practices to particular customers. Other issues are of legal and organizational nature.

4.1.3 Collaboration across tool boundaries

Traceability across organizational borders is especially difficult because of the diversity of tools used in practice. In total, 14 of our 24 interviewees mentioned this issue. Insufficient tool integration complicates creating trace links between artifacts stored in different tools.

A (workaround) solution for this issue is the manual mapping of IDs, e.g., in Case 12 or Case 5. One solution used in Case 11 is to use code comments to reference requirements specifications with their version numbers.

4.2 Common goals and responsibilities

Responsibilities of individuals need to be coordinated, and the benefit of traceability has to be seen. We found this to be challenging in practice, for reasons explained in the following.

4.2.1 Benefits of traceability

Involved stakeholders need to be aware of the benefits of traceability to use it. When creating a trace link, the creator knows in what ways the source and target artifacts are connected. As a development manager from Case 1 states, *"it is just an overhead"* if the creator does not reflect on the future benefit of trace links but only focuses on the effort when creating them. Because of this, an application engineer from Case 8 considers it one major challenge *"to make people understand the benefit."*

The perceived overhead of traceability management is even higher when organizational or tool boundaries are involved. Several interviewees expressed the need for finding a strong reason to make developers accept the potentially high effort. The cost–benefit balance is important in all cases.

Throughout our study, we found only limited tool support for traceability management. Consequently, the use of traceability is difficult even if trace links are captured using a tool:

> But that is the part where we need a better tool than we have now. Because you...All the information is there but it's kind of hard to get a big picture, an overview.
> (a development manager from Case 1)

Better tool support for using trace links would make the benefits of traceability more obvious. In Sect. 6.1, we presented the tasks for which trace links are used in practice. As mentioned before, many of these tasks are only partially supported by tools.

Another complicating factor mentioned by many interviewees is that the creators of trace links are a different group than the stakeholders using the links. For instance, to allow a project manager from Case 2 to track the project's progress, trace links between implementation tasks and source code commits need to be created by developers. Other cases confirm this point:

> In our case, the creators of the links, so the developers, actually complain a lot that they have to think of traceability. The people who really use it are support persons or project managers. So people from another team.
> (a change leader from Case 5)

The attitude of stakeholders is of great importance:

> Some people say 'I don't care, [...] I'm just interested in doing my work, why do you bother me?'
> (an application engineer from Case 8)

It can be concluded that stakeholders have to see the benefit of traceability to accept the overhead of collaboratively working on them. Our findings suggest that this is

particularly difficult when effort and benefit are so unevenly distributed over different roles.

4.2.2 Managing responsibilities

The organization of responsibilities for traceability management is done differently in the industrial cases. In the majority of the cases, a complete team is responsible for it. Traceability management is often ensured by checklists. Interviewees from four cases stated that traceability management needs to be handled as part of the daily work of each stakeholders.

Tool support strongly influences the effort of traceability management. Case 5 uses a traceability matrix in a spreadsheet where trace links are managed. Keeping the spreadsheets up to date is a full-time job for one person. In Case 15, a quality manager performs frequent reviews of trace links. If needed, this person contacts other responsible people to update the links.

Traceability management was compared to the task of documentation by several interviewees. There is a higher priority assigned to tasks that have direct business value, and traceability management is often neglected if its value is not obvious to the stakeholders, as stated by a product manager from Case 10. Managing responsibilities is highly connected to (perceived) *benefits of traceability* (see Sect. 4.2.1): People are more likely to assume their responsibility if they see the benefits.

It should be noted that current practices are often only used within individual organizational groups or disciplines. In many cases, there exist no concepts for traceability across boundaries and responsibilities on a higher organizational level:

> The testing team does it, because they [...] care about which requirements they are covering. But it's not clear to me if somebody else really takes the responsibility for traceability in the company.
>
> (a quality analyst from Case 4)

As a software architect from Case 7 stated, especially links between disciplinary borders and responsibilities to maintain them are difficult to handle:

> So it's also an issue that there are not any persons responsible for the big picture how links are working. There are people responsible for the breakdown of requirements and there are people responsible for the software. In these boundaries not...
>
> (a software architect from Case 7)

Several interviewees stated that better tool support would facilitate this and allow more roles to see the benefit and use of trace links, for instance, by providing a visual overview of connected artifacts.

4.3 Collaborative trace link maintenance

Traceability maintenance is highly relevant to ensure high trace link quality. In this section, we present challenges related to traceability maintenance and collaboration.

4.3.1 Trace link quality

The cost–benefit balance of traceability management is of especially high relevance in collaborative environments, as described in Sect. 4.2.1. Leveraging traceability requires both developers and managers to understand the use and benefit. We found that trace link quality is essential for the trace links to be used, and if trace links are used they are likely to be maintained in a collaborative way.

> Because you need to have trust in the data. You need it to be on the level that you think it's worthwhile looking into traceability to get some benefit. If you don't trust it, then it's not used. And if it's not used, then you don't improve it.
>
> (Chief Technical Officer from Case 8)

There exist several notions of quality and relevant aspects. For instance, a system software architect from Case 14 describes it as a "difficult question" when being asked to define trace link quality. Both the definition and measurement of trace link quality are difficult in practice. Our interviewees considered especially incorrect or missing links between artifacts as problematic.

For instance, the problem with incorrect trace links was described as follows:

> Otherwise if you try to follow up on something, then you're going to get utterly confused if you see that this doesn't belong together at all. I [...] prefer if there is no link instead of there being a wrong link. Because then you have to evaluate that as well, is the link correct or not? But if the link is not there, then I can search in some other way to find out what I want to know.
>
> (a team leader from Case 2)

When discussing the topic of automatic trace link creation, the interviewees' reactions were very hesitant. Whereas we could not observe a major difference between projects of different durations and sizes, we found that there are differences between domains and the importance of having correct trace links. Our interviewees, especially those developing safety-critical systems, e.g., in the automotive domain, described it as essential to have correct and complete trace links. Interviewees from the software development domain were interested in using automatic algorithms to create trace

links—however, rather *"as an input for a final manual step,"* as mentioned by the Chief Technical Officer from Case 8.

An interesting observation was that the notion of correctness changes if one considers versioned trace links. A software architect from Case 7 states that correct trace links become *"outdated"* in case the connected artifacts change rather than *"automatically incorrect."* However, to maintain trace link quality, it is still important to consider change propagation and trace link maintenance, as discussed in the next subsection.

4.3.2 Change propagation and notification

In case one of the connected artifacts changes, one needs to assess which implications the change has on connected artifacts and the trace link. If a versioning solution is used, the trace link needs to be updated to connect the new versions of the artifacts (or deleted if the artifacts should not be in relation anymore).

Many interviewees stated that trace links most commonly become incorrect if an artifact is updated and the trace links or connected artifacts are not changed accordingly. Insufficient notifications are one way how incorrect links are introduced:

> If you change the signal, you won't get any information about what requirements linked to it. That's probably where we introduce incorrect links.
>
> (a software architect from Case 7)

It can happen that a whole chain of artifacts and trace links are affected by a change—for instance, when a requirement is changed. This usually affects different project phases, as changes need to be analyzed, designed, implemented, and tested. In addition, different departments are also affected because a change can affect systems engineering concerns, as well as mechatronics and software engineering. Therefore, change propagation and traceability maintenance are a collaborative task that must be supported by organization-wide traceability.

We found that trace link maintenance is addressed in different ways. Case 14 defined an official workflow to handle changes when an artifact is modified. The workflow includes informing all affected stakeholders about the change and its implication, in order to ensure that artifacts and trace links are updated. Several interviewees from other cases reported problems due to missing notifications:

> We have exactly this problem: someone changes something at the top and nobody notices it. Or not everybody [who should know] notices it.
>
> (Head of Test Management from Case 15)

When asked about possible solutions to this problem, our interviewees stressed the importance of collaboration between teams:

> You have to talk about it. There are frequent meetings with [involved stakeholders] to discuss these changes and their impact. And each of them has to follow up on his or her tasks.
>
> (Head of Software Quality Assurance from Case 12)

The issue of traceability maintenance is especially difficult when working in *"distributed teams that do not meet in the break room every day,"* as the Head of the Project Management Office from Case 12 states.

5 Traceability: an enabler for collaboration

Our research suggests that traceability provides opportunities for collaboration. In this section, we investigate *RQ2: How can traceability management support collaboration?* We asked our interviewees to what extent traceability management and collaboration influenced each other and what collaboration practices with internal and external partners looked like. As shown in Fig. 1, four main opportunities emerged: easier communication in distributed environments and interdisciplinary engineering to collaborate across boundaries (Sect. 5.1), explicit documentation of decisions to establish common goals and responsibility (Sect. 5.2), and creation of trace links to receive information to support trace link maintenance (Sect. 5.3).

5.1 Opportunities: collaboration across boundaries

We identified two opportunities with collaboratively addressing traceability across boundaries: (1) easier communication in distributed environments (Sect. 5.1.1), and (2) interdisciplinary engineering (Sect. 5.1.2).

5.1.1 Easier communication in distributed environments

Communication in distributed environments with different stakeholders can profit from being able to collaborate in the context of trace links.

A product manager from Case 10 arrived at the following conclusions when considering collaboration features in a tool:

> But you should not forget that people work closely together in software projects. Today, it's globally, [...] distributed and in different time zones. And of course you have the possibility to send e-mails, but the plethora of emails that you receive nowadays...So it makes sense if you have close collaboration in distrib-

> uted teams at distributed locations. Then having such ways to communicate that are relatively easy might be appealing.
>
> (a product manager from Case 10)

According to our interviewees, communication in the context of a set of specific trace links can be facilitated by a collaborative traceability tool that allows maintaining all pertinent information in a shared space, where it can be easily accessed by all collaborators in a meaningful context.

5.1.2 Interdisciplinary engineering

Software engineers hardly ever work in isolation. To integrate all aspects from different disciplines consistently in one end product, a successful coordination with systems and hardware engineers is crucial. Many of our interviewees were mainly concerned with software—however, also systems engineering concerns are relevant in some cases. Trace links can support propagating changes from one discipline to all other affected disciplines, as reported by a system software architect from Case 14:

> Traceability can help at the point where you have changes in one domain. [...] For example, [if] I have a software port...If it is inside the software I can change it as I want. But if it is the hardware-software interface, someone else has to notice. And to get exactly this information—which ports I can change without consultation and which I can't—that's what traceability is important for.
>
> (a system software architect from Case 14)

The interviewee implies that traceability can also help to determine the effects of changes and whether other disciplines must be notified, as described in Sect. 4.3.2.

5.2 Explicit documentation of decisions

One concern raised in the interviews was the usage of a common platform with external companies and its potential impact on collaboration. In this context, two interviewees described situations where inadequate traceability had a negative effect on collaboration with external partners.

> [Using the same system as our partners] would make sense to get the end to end traceability here. As I said, yesterday we had an issue with that. Everyone had always said, 'everything works fine, the delivery will be on time'. And then three days before it wasn't as good as we thought, 'ah, so you meant it that way? I didn't know that at all.' [...] But with traceability you can see that from the start, have they understood it this way, do they know that we have [these connections to other artifacts].
>
> (Head of Project Management Office from Case 12)

Trace links can, at least to a certain extent, also be used to ensure that a team is aware of certain dependencies and can refer to a common source of documentation for such dependencies. A project manager from Case 1 suggested to record requirements and their trace links together with all involved stakeholders during the requirements engineering phase. The interviewee stated that such explicit connections help to make decisions in a team.

An idea for future tooling was to include a voting feature. A voting feature would allow users to voice their opinions regarding trace links, indicating incorrect/suspicious links or that they agree with artifacts being connected with a trace link. Interviewees from Case 15 state how they would use a voting feature in their reviews so that they can more easily comment on trace links and approve or disapprove the latest changes.

> There you can see who said 'okay' or 'not okay' and what they commented on, what they did not agree with.
>
> (Head of Test Management from Case 15)

This interviewee suggests that a voting feature can support decision making in a team and help stakeholders to jointly improve the quality of trace links. According to another interviewee, this could help during elicitation and prioritization, e.g., to record potential dependencies. Such suggested links are a valuable input for trace link management, as, for example, the Chief Technical Officer from Case 8 suggests.

5.3 Creating trace links to receive information

Another idea from our interviewees was to react to artifact changes by sending the person(s) responsible for connected artifacts an automatic notification and request to review all affected trace links.

Such a change notification feature was indeed regarded as being beneficial:

> When the requirements engineer changes something and I notice it through the chain [of trace links] then this facilitates collaboration. Because if one forgets to tell the others about a change then nobody notices it. And [change notification] would have created a big advantage.
>
> (Head of Software Quality Assurance from Case 12)

Multiple interviewees mentioned the potential of collaborating via trace links. Creating links can be viewed in this

Table 2 Traceability management approaches, their characteristics, and case contexts

Traceability management approach	Requirements-centered	Developer-driven	Mixed
Used in	Cases 3–6, 10, 13–15	Cases 1, 2, 8, 11, 12	Cases 7 and 9
Countries of cases	5 German	2 German	0 German
	3 Swedish	3 Swedish	2 Swedish
Focus of tracing	Requirements to sub-requirements, requirements to test cases	Implementation tasks/issues to source code commits	Both
Drivers to manage traceability	Compliance with regulations and standards	Observed benefit by developers	Benefit in several steps of the development
Development paradigm	Officially V-model, with upcoming agile trends	Agile (except for one case)	V-model and agile
Project sizes and distribution	Typically larger, distributed teams	Smaller, local teams	Varying. Developer-driven approaches often only used in a smaller team
Approaches to collaboration	Formal, supported by documents/ specifications	Informal, face-to-face communication	Mixed
Collaboration with external stakeholders	Strong, often customer–supplier relationships	Less relevant	Less relevant
Used tools	Mostly requirements management tools	Configuration management system/ issue tracker	Both

manner as a means of ensuring that changes are communicated correctly: A trace link between a source and target artifact ensures that the person responsible for the target artifact gets notified when the source artifact changes.

6 Traceability management approaches

This section discusses and answers *RQ3: How does collaboration relate to different approaches of traceability management?*

Collaboration relates to different aspects of traceability, with what goals it is used, and with what underlying approaches it is managed. On the one hand, we found that the goals behind establishing and managing traceability affect collaboration (Sect. 6.1). On the other hand, we identified three basic approaches to collaborative traceability management and concluded that they relate to collaborative aspects in various ways. In this section, we describe the three approaches in detail: requirements-centered (Sect. 6.2), developer-driven traceability management (Sect. 6.3), and mixed approaches (Sect. 6.4). Table 2 gives an overview of the three approaches with their characteristics and contexts. For each approach, we name the cases using it, what the focus of tracing was, and what the drivers to manage traceability are. Besides, we describe the development paradigms that are used in the approaches and the project sizes and distributions, which greatly influence the context of traceability management. Finally, Table 2 shows the approaches to collaboration, collaboration with external stakeholders, and the used tools.

6.1 Traceability goals' relation to collaboration

When discussing traceability management approaches, we created a list of top five goals behind traceability management. In the following, we list them with the relation to collaborative aspects and a short description of how trace links are used in the respective context. An overview of the cases and how they use each of these goals is shown in Table 3.

1. Tracking the project's progress: This purpose is a common theme in the traceability literature (e.g., [8, 41]) and includes two scenarios. In the first, trace links between commits and issues—usually established via pointers in the commit message—can be used to identify issues currently under development and which status the implementation has reached. In the second scenario, trace links between requirements, tests, and test executions are used to expose tests that have failed for certain requirements and where additional work is needed. This goal is typically followed by stakeholders interested in project management, requirement engineers, or testers, and can help to identify how artifacts relate.
2. Troubleshooting: Trace links between test cases and specific artifacts (code, classes, state machines) support developers in identifying the causes for failed tests. It is particularly helpful when multiple developers need to collaborate to understand each other's artifacts and test cases.
3. Coverage analysis: A traceability matrix can show which requirements are covered by test cases and which requirements still lack sufficient testing. Similar to the other goals, this is typically pursued by a group of stake-

Table 3 Goals and uses in each case

Domain and Case no.	Goal 1: Track project progress	Goal 2: Troubleshooting	Goal 3: Coverage analysis	Goal 4: Change impact analysis	Goal 5: Generating artifacts
Embedded					
Case 1	X	X	X	X	X
Electrical equipment					
Case 2	X	X			
Industrial automation					
Case 9	X	X		X	
Automotive					
Case 3			X		
Case 4			X		
Case 6	X	X			
Case 7	X		X	X	X
Case 10	X	X	X	X	X
Case 14	X	X	X	X	X
Telecommunications					
Case 5	X	X		X	
Case 13			X		
Software development					
Case 8	X	X	X	X	X
Case 15	X	X	X		X
IT services					
Case 11	X			X	X
Banking self-service automation					
Case 12	X		X	X	X

holders aiming for a high level of coverage, and possibly related to tracking the project's progress.

4. Change impact analysis: Trace links establish relationships between the architecture and requirements. If a requirement is changed, trace links can reveal which architectural elements are affected by this change. However, with the current tool support, it is often necessary to manually traverse the data for this purpose, as stated by a system software architect from Case 14. As these artifacts are typically created and used by different people, change impact analysis requires a collaborative effort of stakeholders who communicate about their changes.
5. Generating artifacts, e.g., release notes/status reports: During release planning, milestones are assigned to issues, thus introducing trace links between the milestones and the issues. A milestone report can then show which issues are currently receiving commits, as well as use trace links connected to tests to show which issues have been fully implemented and have sufficient test coverage. This is typically used to communicate about the status of the software or system to be developed.

It can be seen that all goals are used in a collaborative setting including several individuals that create, maintain, and use trace links.

6.2 Requirements-centered traceability management

We categorized eight of 15 cases as relying on a requirements-centered traceability management approach, characterized by a strong focus on requirements management when it comes to the use of traceability.

With the exception of Case 6 and Case 13, all of the requirements-centered cases rely on a dedicated requirements management tool for organizing and structuring the requirements. The tool most commonly used was IBM Rational DOORS[3] (especially in automotive cases), used in four of the eight requirements-centered cases. Among other features, the tool allows managing requirements and test cases on different levels of granularity. It not only supports traceability between these artifacts, but also between other artifacts stored in external tools through interfaces such as OSLC.

[3] http://www-03.ibm.com/software/products/en/ratidoor.

An information management tool is also used in Case 6, storing requirements, analysis and design models, test cases, and trace links between requirements and other artifacts.

Integration and system test cases are usually stored in a connected test tool and linked to the requirements. In Case 6 and Case 14, requirements are also linked to analysis and design models. Different stakeholders are involved, as described by a system software architect from Case 14:

> The system architect creates links to the system requirements, the software architect creates links to software requirements, and the component developers to the software architecture.
>
> (a system software architect from Case 14)

In Case 13, connections between requirements and design artifacts of a measurement system are recorded. This is formally done in spreadsheet files and documents, following a strict process.

The usage of traceability management in requirements-centered approaches is characterized by fixed, plan-driven processes, often officially following the V-model. Many of these cases are part of large organizations or represent situations in which several companies work together. Often, customer–supplier relationships and collaboration with external organizations are relevant in these cases. This goes along with more organized processes and responsibilities.

In Case 15, for example, formal reviews are conducted before a milestone is completed, also making sure that trace links are set. In this particular case, stakeholders are invited for the review using a feature in DOORS that sends out e-mails automatically.

It should be noted that the majority of the requirements-centered cases belong to the automotive domain where safety criticality and OEM–supplier relationships require formal specifications and organized ways of working.

Interviewees following the requirements-centered approach stated that communication is typically handled via e-mail and document exchange. Meetings are often held using telephone or video conference systems, as well as regular face-to-face meetings when possible. However, generally, a rather formal approach to collaboration is followed.

In most of these cases, traceability was established in a top-down fashion from management side, motivated by quality or safety standards that the company wants to fulfill. We observed that the more relevant safety or quality certifications are for a company, the more formally defined are the processes for requirements-centered traceability management. It also relates to the formal way of collaborating in these cases. Trace links are typically created by project managers, test managers, software architects, and/or the responsible developers for features.

6.3 Developer-driven traceability management

Besides requirements-centered traceability management, we identified more developer-driven, ad hoc approaches, used in five of 15 cases.

In most of these cases, the approach involves the use of an application lifecycle management or software configuration management system. Team Foundation Server[4] is used in two of five cases. In many cases, bug or project tracking tools such as JIRA[5] together with a version control system like Git[6] are used, e.g., to manage implementation tasks or reported defects of a project, collectively referred to as "issues" in the following.

The motivation for traceability is the potential to relate changes in the code to the connected requirement or issue. This is particularly used during troubleshooting. It is also used to track the status of the project.

Cases following developer-driven approaches typically drive their development based on issues and connect source code to them, e.g., by stating the issue ID in commit messages. This is one important way of creating trace links and keeping track of the development state, as emphasized by the Chief Technical Officer from Case 8:

> Issues represent some kind of unresolved problem. And to have them as issues, it gives a clear statement of what remains to be done. So that really drives the development. Doing it in an unstructured way wouldn't be possible.
>
> (Chief Technical Officer from Case 8)

In Case 11, trace links are created inside the code, as source code comments. This creation is a workaround solution as the developers missed the functionality to link from the source code to the requirements. In the future, it is planned to use a software configuration management system to handle the creation of trace links as in the other cases following the developer-driven approach. References to the respective requirements specification version are made.

The creators of trace links in developer-driven traceability management are typically developers with a focus on the actual software implementation. The approach is often only used in the development departments in an organization, sometimes only by a small team.

Communication is often handled in face-to-face conversations, (weekly) meetings, and via e-mail exchange. Compared to the requirements-centered approach, traceability is not discussed as much in conversations and meeting, and is not considered an essential organizational goal. Instead,

[4] https://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx.

[5] https://www.atlassian.com/software/jira.

[6] https://git-scm.com/.

informal ways of collaborating are frequently used, which makes it more difficult for stakeholders to see the need for formal documentation.

Developer-driven traceability management is especially common in cases with an agile paradigm. Three of the cases following the developer-driven approach use the agile paradigm, and the fourth agile case in our data follows a mix of approaches, as shown in Table 2. In these cases, the software configuration management system is also used to plan the work. Collaboration with other organizations is less relevant in these cases. For this reason, the natural collaboration approach is rather informal and focused on the needs of developers.

6.4 Mixed approaches

In two cases, there are attempts to handle traceability management in a structured, top-down way, and at the same time, bottom-up approaches are pursued by the developers. Note that while this suggests a best-of-both-worlds approach, we could not observe a systematic endeavor to combine tracing information from both parts.

In Case 9, there exists a document-based, requirements-centered approach to link features and test cases using ID mapping, i.e., the IDs of the features appear in the test case names. One interviewee states that the trace links exist but that it requires manual work to identify which artifacts are connected:

> I can do it now through the links in the documents. So I can take my test report and see 'this is this release of the test framework', and I can go to my documentation to see what features we are testing, and I can link it. But it's still manual.
>
> (a software quality manager from Case 9)

Apart from that, Case 9 also uses more developer-driven approaches, linking bugs or defects to the respective code commits in which they are fixed.

In Case 7, test cases and design artifacts are linked to requirements in an information management tool. Additionally, as in many other cases, an issue tracking tool is used by the development department and trace links from issues to source code commits in a version control system are created. Apart from that, there are other tools used in different parts of the organization. A software architect from Case 7 points out that this is problematic in practice:

> The biggest discrepancy between the ideal tool and what we have today is probably that we have many tools today, and not one. If it were one tool [...] then you would probably get more [analysis features] automatically.
>
> (a software architect from Case 7)

Table 4 Cases with their level of agility and rigor of following development paradigms

	Agile	Mix of paradigms	Plan-driven
Rigorous	Case 9 (Mix)		Case 14 (Req)
	Case 12 (Dev)		Case 10 (Req)
			Case 4 (Req)
			Case 3 (Req)
			Case 15 (Req)
Balanced	Case 2 (Dev)	Case 11 (Dev)	Case 6 (Req)
		Case 13 (Req)	Case 7 (Mix)
Lenient	Case 8 (Dev)	Case 1 (Dev)	
		Case 5 (Req)	

Different tools are in use, and the focus of tracing varies depending on the stakeholder. Regulations and certifications do not play as much of a role as in the requirements-centered approaches. This approach is more focused on the benefit of traceability in different steps of the development. The development is either agile or V-model based with agile paradigms established in a bottom-up fashion. In mixed approaches, it should be noted that typically a smaller development team introduces developer-driven traceability management. At the same time, requirements-centered approaches are established in a more top-down fashion.

Depending on the discussed issue, various means of communication are used. More formal communication is used for the requirements-centered part, and informal collaboration within developer teams. Our cases focus on direct conversations wherever possible, but also exchange e-mails. Traceability-related topics are typically not discussed a lot, given that they are not as central for the cases' organizational goals as in the requirements-centered approaches. The collaboration with external stakeholders is less relevant in these cases.

7 Influence of development effort characteristics

This section provides answers to *RQ4: What characteristics of the development effort influence traceability management and collaboration?*

We found that several characteristics play a role in the context of traceability management and collaboration. Most prominently, culture, organization, and processes influence how traceability is managed and how collaboration is approached. These aspects are connected and were an underlying theme in all interviews. Also the way individuals collaborate depends on these factors. Underlying development paradigms and processes mirror the organizational structure, and the strictness or rigor of following

them allows us to deduce cultural aspects. For this reason, we discuss underlying development paradigms and the rigor of following them in different cases.

Table 4 shows the cases of this study with the level of agility and the rigor of following their development paradigms. If a case company uses an agile paradigm and the interviewees describe their way of working accordingly, the case is located in the "Agile" column. Cases following a plan-driven approach (most commonly the V-model) are located in the column labeled "Plan-driven." In the column "Mix of paradigms," cases are located whose development is characterized by a mixture of agile and plan-driven elements.

In the rows, we categorize the cases with respect to their level of rigor, indicating how strictly a development paradigm is defined, communicated, and executed in an organization. If a company has a strict definition of the development paradigm and the practices were confirmed independently by all interviewees, the case is labeled "Rigorous." The amount and types of trace links used are irrelevant, but the intended paradigm and practices need to be followed for a case to be considered rigorous. If interviewees state that the development paradigm is not followed strictly, or we found contradicting statements about how the paradigms are communicated and followed, the case is categorized as having a medium level of rigor. Cases ranked as "Lenient" reported on inconsistent practices and the lack of an overall strategy or traceability concept. Of course, the sample of the interviewees and their openness when answering questions influence the way we ranked companies. Data (source) triangulation by interviewing several individuals helped to mitigate this bias.

We found that whereas (almost) all cases have an officially defined development paradigm, the extent to which it is followed and communicated differs a lot. This also impacts traceability management, but is not limited to it. In parentheses, we indicate the used traceability management approaches (see Sect. 6)—requirements-centered (Req), developer-driven (Dev), or mixed approaches (Mix).

Answering RQ4, our findings suggest that cultural and organizational aspects are coupled with development paradigms. Both the level of agility and the level of rigor greatly impact which traceability practices and what collaboration approaches are applied. We elaborate on this in the following, describing rigorous cases (Sect. 7.1) and cases with less rigor (Sect. 7.2).

7.1 Rigorous cases

There are several rigorous cases—both related to plan-driven approaches and to agile approaches. We describe both groups in the following.

Traditionally, plan-driven cases work in projects and use traceability to ensure that requirements are implemented and tested. Another motivator is safety and quality standards that require strict processes. Due to these processes, also collaboration is approached in a rather formal way, using specifications and formal reviews. Traceability is required to demonstrate how requirements are broken down and fulfilled later on:

> And as soon as we have requirements in place and start developing, we should make the developers somehow relate to requirements, to functionalities. It definitely favors the project if those connections are made.
>
> (a quality analyst from Case 4)

The majority of the plan-driven cases operate in the automotive domain. Several interviewees mentioned that a transition toward agile is being considered. Several contextual factors complicate this transition, e.g., the traditional supplier–OEM relationships in the automotive domain and the multitude of involved disciplines.

Some of the plan-driven cases use a significant amount of reviews and assessments to ensure correct trace links or by more clearly defining and controlling responsibilities. Less rigorous cases allowed processes and responsibilities to emerge bottom-up. The only reported case in which end-to-end traceability was achieved was a plan-driven rigorous approach.

Besides rigorous plan-driven cases, we also found cases following rigorous agile paradigms. Rigorous agile cases are characterized by a strong focus on quality, a global definition of agile values, principles, and practices beyond individual development teams, and the purposeful use of traceability. The importance of face-to-face communication as an agile principle influences how stakeholders collaborate. The introduction of the agile paradigm is typically not done in an ad hoc way, but in a well-planned, long-term transition process.

Most cases following a rigorous agile paradigm create and use trace links between themes, epics, and user stories, as well as to test cases. Traceability is actively considered in rigorous agile cases:

> Yes, the traceability definitely got better [during the transition towards agile]. Especially because before, we had big documents that were written as text, and now we have atomic requirements.
>
> (Head of Software Quality Assurance from Case 12)

We found that the goals of traceability in rigorous agile cases are slightly different than in plan-driven cases. Safety and quality standards play less of a role. Interviewees from all rigorous agile cases stated that "the whole team" should be the users of traceability.

One commonality in rigorous agile cases is that interviewees describe the trace link quality as "good, but not good enough" (a software quality manager from Case 9). The interviewees showed interest in improving their traceability approaches. They all state that tool support is missing to leverage traceability in a better way (e.g., a software quality manager from Case 9 and the Head of the Project Management Office from Case 12).

7.2 Balanced or lenient cases

We observed that a number of cases followed a less rigorous approach to traceability management and collaboration. Many of the balanced or lenient cases are currently undergoing a transition from plan-driven to agile paradigms. It was mentioned by several interviewees that the organization requires V-model processes, whereas software development teams or departments internally work in more agile ways. These cases reported difficulties with trace link quality and maintenance. Trace links are mainly used for troubleshooting, but "links are not always there when they should be there," as mentioned by a team leader from Case 2. The interviewees reported that traceability is typically established in a bottom-up way inside the development department, but the scope is too limited to be beneficial for the company. From a collaborative perspective, it could be observed that collaboration typically works well in a smaller team, but organizational boundaries are not crossed to exchange knowledge in a satisfying way.

A developer from Case 6 describes challenges with inconsistency and change propagation being very present:

> So right now, the documents are saying something and the [product] is doing something else. It happens a lot of times. Because there are, somewhere in the process, someone changes and doesn't go back. There is no feedback loop. It's more like a V-model. There is no connection.
>
> (a developer from Case 6)

There were several companies that are currently in an active transition toward more rigorous agile paradigms, which often explains the above-mentioned inconsistencies.

When asking interviewees what they would change in their traceability management practices if they could start from scratch, many stated that they would prefer to have a more rigorous approach:

> But if I could start from scratch, I would do some things more thoroughly. First of all, to coordinate the rules of traceability and require people to follow them. If you grow bottom-up, you can do that. If you have a huge amount of code and then one percent that you create trace links for, then it is questionable how much benefit you get.
>
> (the Head of Development from Case 11)

In some companies, a long-term transition is performed in a coordinated way, involving stakeholders from different departments and aiming for a holistic traceability solution. Interviewees from Case 12 stated that it can be problematic to run initiatives on a minor level with limited influence on the rest of the organization, which also complicates traceability.

8 Discussion

This section discusses our research findings and implications for researchers and practitioners.

To summarize our findings with related work, Table 5 shows an overview of themes and topics. We also name the cases in which the themes were discussed most prominently.

Based on the discussion of our findings, we arrived at practical principles for collaborative traceability management. We motivate the principles in Sects. 8.1–8.4, where we discuss each of the research questions.

8.1 Challenges

We answer RQ1 based on our findings: What are practitioners' challenges with collaboration in traceability management? We identified three groups challenges that we summarize and discuss in the following:

8.1.1 Collaboration across boundaries

Our findings indicate that collaboration across team and discipline boundaries is challenging in practice. The issue of collaborating in distributed organizations has been confirmed by related studies. Sinha et al. [54] mentioned challenges related to globally distributed requirements management, many of which we found to also hold for traceability management: There is an impact not only because of geographic separation, but also because of different disciplines and backgrounds, used tools, and organizational separation of departments. This separation is partly due to separate groups of traceability stakeholders with different viewpoints having unaligned goals—for instance, between upper management and engineers [23]. Other issues found by related work are organizational problems, e.g., politics or lack of training [23].

Kirova et al. [24] analyzed the cost of traceability and the challenges of tool boundaries and heterogeneous tools. Our findings confirm the need for better support for analysis

Table 5 Overview of themes in relation to case companies and related work

Theme	Topic	Main relevant cases	Related literature
RQ1—Challenges: Managing traceability collaboratively			
Collaboration across boundaries	Collaboration with other departments	2, 3, 6, 9, 10, 12	[23, 54]
	Collaboration with external organizations	1, 2, 8, 7, 11, 12, 14	
	Collaboration across tool boundaries	In 14 (of 15) cases	[3, 17, 24]
Common goals and responsibilities	Benefits of traceability	1, 2, 5, 8	[2, 5, 23, 37, 49]
	Managing responsibilities	1, 4, 5, 7, 8, 9, 10, 14, 15	[13, 18]
Collaborative maintenance of trace links	Trace link quality	2, 3, 7, 8, 12, 14	[43, 57]
	Change propagation and notification	6, 7, 12, 14, 15	[52]
RQ2—Traceability: An enabler for collaboration			
Easier communication in distributed environments		10, 15	[17, 54]
Explicit documentation of decisions		1, 7, 12, 15	[40]
Creating trace links to receive information		7, 12	[21, 51]
Interdisciplinary engineering		14	[22, 26]
RQ3—Traceability management approaches			
Requirement-centered		3, 4, 5, 6, 10, 13, 14, 15	[6, 25, 30]
Developer-driven		1, 2, 8, 11, 12	[30]
Mixed		9, 7	[6]
RQ4—Influences of development effort characteristics			
Rigorous		$3-4, 9-10, 12, 14-15$	[6, 7, 16, 17, 30]
Balanced or lenient		$1, 2, 5-8, 11, 13$	[25, 44]

of trace links and flexibility when it comes to diverse paradigms and tools.

Asuncion et al. [3] addressed this challenge by centering the data exchange of their software traceability solution around a shared database accessible by all tools. However, when different companies have to collaborate this can be infeasible due to legal issues, as we described in Sect. 4.1.2.

8.1.2 Common goals and responsibilities

We observed a lack of common goals and responsibilities, which is strongly related to the lack of benefits of traceability. To justify the effort of traceability management, it is relevant but often challenging to convey the benefit of traceability to involved stakeholders.

We found that the creators of trace links are often not the users of information. It is difficult to motivate stakeholders to invest in trace link quality if the information needs and goals of traceability are unclear. This is what our first principle addresses: (P1) *Put stakeholders' information needs and goals of traceability at the center.*

The alignment of viewpoints and goals of traceability stakeholders was also found to be challenging by Kannenberg and Saiedian [23]. In fact, we found that the core of the issue is that the people leveraging trace links are typically not the stakeholders creating the links.

Our findings confirm the problems with traceability mentioned already in 1994 by Gotel and Finkelstein [18]: They found that invisible responsibilities complicate the exchange of information among parties. We identified the issue that responsibilities for traceability are often not clearly defined when organizational boundaries need to be crossed.

Arkley and Riddle [2] also stressed the lack of perceived benefit of traceability to stakeholders, especially to the trace link creators. Another issue was that information in the traceability tool had to be entered multiple times. These experiences support our findings that tool support impacts how traceability is perceived and used.

Panis [37] discussed several challenges related to traceability degradation, the benefits of traceability, and stakeholders' motivation. He concluded that traceability needs to be "automatically visible for engineers as part of their daily work" to motivate them to improve trace link quality. Similarly, our findings suggest that beyond seeing traceability in their daily work, stakeholders are much more likely to improve trace link quality, when they see the benefits of traceability.

We interpret collaborative traceability as a special case of experience and knowledge management [49]. From this perspective, it is clear that an experience bearer (e.g., a stakeholder with knowledge about a dependency between requirements) is more likely to share knowledge (e.g., by fixing a wrong trace link), if the benefit of doing so exceeds the effort. In this context, Averbakh [5] distinguished lightweight approaches, which aim at reducing effort, from heavyweight approaches, which aim at maximizing benefit.

Averbakh recommended minimizing the time required to share knowledge (i.e., maintain traceability) and shift effort away from the experience bearer to others. The latter is also crucial for traceability: Roles that possess valuable knowledge to be captured in trace links (e.g., developers) are often not the ones that have the main benefit. For this reason, we arrive at the following principle: (P2) *Balance the effort and benefit of traceability management per role.*

8.1.3 Collaborative trace link maintenance

Trace links and trace link quality are often directly affected by changes of connected artifacts. Therefore, trace links and artifacts need to be maintained, which can be challenging if collaboration between stakeholders is insufficient. In this context, we found especially that a lack of change notification and propagation can impede traceability maintenance. Besides trace link quality, it also negatively affects collaboration across organizational, discipline, and tool borders. For this reason, we specify the following principle: (P3) *Enable change propagation and notification across boundaries.*

Sengupta et al. [52] found similar challenges with the propagation and management of requirement changes in general. Requirements changes typically have an effect on design and coding, which is where traceability could be leveraged as a facilitator. We expect suitable change notification features to support not only trace link maintenance, but the general issue of change propagation.

The topic of trace link quality has been addressed by Rempel and Mäder's [43] model to assess requirements traceability. Our findings indicate that it is central to analyze quality of trace links. Ongoing research investigates how the quality of manually created and potentially untrustworthy trace links can be improved (cf., e.g., [57]).

8.2 Effects of traceability management on collaboration

In the following, we discuss our results for RQ2: How can traceability management support collaboration?

We found that collaborative issues can be mitigated with adequate traceability management. This requires that collaborators have common goals, and see that the perceived effort of traceability is exceeded by the benefit of overcoming such boundaries. The issue of tool boundaries could be facilitated by tool suppliers that move toward common standards such as OSLC.[7] Whereas tool boundaries and the lack of applicable tool support play a role in practice, they are not the only complicating factors to be blamed. More severe is the issue of invisible benefits of traceability which makes it difficult to leverage traceability in practice. Therefore, we see it as imperative to create direct, tangible advantages for engineers that actively maintain trace links by allowing them, e.g., to more easily identify the impacts of change or to engage other members of their distributed team, and thus ensure that a positive cost–benefit balance is perceived by all stakeholders. Our findings indicate that traceability can be used to enable collaboration and facilitate communication and knowledge management, especially in distributed teams.

Our findings indicate that traceability management can improve collaboration in the following four ways:

8.2.1 Easier communication in distributed environments

Based on an evaluation of a distributed requirements management tool, Sinha et al. [54] identified *contextual collaboration* as a useful feature. Contextual collaboration allows stakeholders to initiate conversations that include links to requirements and other artifacts. Our findings confirm that it is indeed beneficial to have better support to collaborate in the context of artifacts or trace links, especially in distributed teams and for collaborative traceability management. It is important to note that users must be comfortable with the provided tool solutions. Leveraging existing communication mechanisms known to the user is thus important [54].

Based on our findings, we conclude that in distributed environments, developers would ideally introduce and maintain useful trace links as a by-product [48] of the necessary project documentation (e.g., by allowing references to specific potential trace links in chat rooms). This would reduce effort, while improved communication would increase the benefit of traceability management. Thus, the effort/benefit ratio would be improved, and developers would be more likely to contribute to traceability management.

8.2.2 The explicit documentation of decisions

Ramesh [40] described that traceability plays a crucial role for knowledge management processes. Among other advantages, trace links can support the documentation of critical design decisions or assumptions.

Our study confirms the need for more explicit documentation of decisions and the potential of traceability to support it. Unclear decisions can block developers. By increasing the explicitness of decision that relate to relevant artifacts, the benefit of traceability is increased for developers, who are then more likely to invest in trace link quality. Traceability is thus an important enabler for documentation and knowledge management.

8.2.3 Creating trace links to receive information

As the suggested change notification feature ensures that responsible stakeholders are informed about relevant changes,

[7] http://open-services.net.

it could actually improve and simplify collaboration. Helming et al. [21] suggested a model-based change awareness approach, identifying which users to notify based on trace links. This was evaluated in a case study and found to be indeed useful in practice. We can confirm that practitioners are interested in such features. With more agility and moving toward continuous integration and deployment, technical dependencies between teams will increasingly appear at a later stage [51]. Thus, we expect that developers will more likely benefit from receiving information about changes, shifting the effort/benefit relationship to their favor and making it more likely that they will contribute to trace link quality.

8.2.4 Interdisciplinary engineering

Dependencies between different parts of a system can be captured explicitly using trace links, helping users to determine which disciplines must be consulted when certain changes are made. Our study suggests that traceability can positively impact collaboration as well as change management in interdisciplinary contexts. This enabler of traceability was only mentioned by one interviewee, a system software architect from Case 14. This theme was emerging from the interview and not part of our interview guide. A potential explanation for the low number of interviewees reporting on this theme could be that the other interviewees' focus laid more on one discipline, rather than systems engineering. Future studies could directly address this theme and further examine the potential of traceability for interdisciplinary engineering.

However, we found that a number of related studies confirm our finding. Königs et al. [26] concluded, based on an analysis of current systems engineering practices, that traceability can provide substantial support for interaction and communication in interdisciplinary scenarios. This claim is supported by investigating two tools supporting traceability management in systems engineering contexts. Similarly, Jaber et al. [22] presented the results of a survey focusing on what kinds of trace links different stakeholders are interested in, and what artifacts need to be connected. Through experiments they demonstrated that these links are useful for supporting maintenance tasks and for fostering collaboration between business and technical stakeholders.

8.3 Traceability management approaches

In the following, we discuss our findings concerning RQ3: How does collaboration relate to different approaches of traceability management?

We found that there exist (1) requirements-centered traceability management, (2) developer-driven traceability management, as well as (3) mixed approaches, combining the former two. As the names suggest, requirements-centered traceability management is focused on requirements as the main artifacts, whereas developer-driven traceability is mainly conducted and leveraged by developers. Organizations typically introduce requirements-centered traceability to assure the quality or safety of their products in a top-down manner. Developer-driven traceability management originates from development teams.

In requirements-centered cases, collaboration is often achieved in a formal setting that helps regulate the interactions between a typically larger team and a customer who is often very involved in the projects. Much of this collaboration is facilitated through a requirements management tool, and both traceability and collaboration are focused on the requirements. In contrast, developer-driven cases are usually found in an environment where a small team communicates informally and face to face and where collaboration with external stakeholders is less relevant. An issue tracker is often used as the main locus of traceability, e.g., between commits and tickets.

These findings indicate that the concrete way that collaborative traceability management is implemented in an organization depends heavily on the established collaboration approaches and the team size. To realize the potential benefits of traceability management on collaboration as presented in Sect. 5, a traceability strategy in a more formal setting needs to be equally formalized with strong tool support and potential enforcement by the tool. On the other hand, in less formal settings it is easier to establish norms and rules regarding collaborative traceability management in an ad hoc fashion within the smaller team. Enforcement can be based on social pressure or a definition of done.

Our findings cannot confirm Cleland-Huang [7]'s finding that trace links between requirements and test cases in agile projects are used. In the agile cases of our case study, traceability of requirements to test cases was not mentioned. To some extent, we can confirm Mäder et al.'s classification [30]: There exist "regulated" traceability users, subcontractors, consultants, and enthusiasts. For instance, the regulated type is in line with the requirements-centered approach, motivated by the need to comply with standards and enforcing processes. Our findings do not suggest that the concrete types of inter-organizational dependencies (working as a consultant or sub-contractor) strongly impacted the way traceability is used. However, collaboration across organizational boundaries did play a role and comes with challenges, as we presented in Sect. 4.1.2.

8.4 Contextual factors' influence

In the following, we summarize and discuss RQ4: What characteristics of the development effort influence traceability management and collaboration?

Similar to earlier studies, we found that traceability management can also be leveraged in agile projects. Based on our data, we could not confirm differences between traceability management in agile contexts that depend on scale, complexity, and safety-criticality that were presented in [7]. This can be connected to the fact that the large-scale, complex, and safety-critical cases in our study came from the automotive domain where the transition to agile paradigms is still in progress. We found that traceability management can be used in both agile and V-model-based contexts. However, our findings indicate that it might be desirable to rather rigorously define and follow traceability management strategies. It is difficult to find a good cost–benefit balance if the practices are followed in an inconsistent way and the quality of trace links drops. We capture this point in the following principle: (P4) *Strive for a rigorous culture with respect to traceability maintenance.*

We identified several cultural, organizational, and process factors that impact how traceability management is conducted. Differences between approaches can be most prominently observed in the development paradigms and the rigor of following them. Our findings suggest that agile and plan-driven cases focus on different artifacts and purposes of traceability: Cases following agile paradigms tend to focus more on development artifacts, whereas plan-driven cases use requirements-centered traceability. It is crucial to analyze the traceability goals and choose and appropriate approach that fits to the purposes. We relate these points to the importance of stakeholders' information needs and goals (P1), and capture them in the following sub-principles: (P1a) *Choose a lightweight, developer-centric approach for goals related to tracking change to code.* (P1b) *Choose a formal, requirements-centric approach for goals related to regulation (e.g., safety or legal requirements).*

Moreover, the trace link quality and maintenance appear to relate to the culture and rigor: Lenient cases and less well-defined paradigms reported difficulties with trace link quality and maintenance. In case trace links are not established in a systematic manner, stakeholders struggled with identifying the benefit and use of traceability, and thus with finding motivation to create trace links.

Our findings are in line with Rempel et al.'s finding that practitioners need to specify traceability goals, and find ways to implement and assess them [44]. In practice, this is often not the case, which results in inconsistencies between the goals of the development paradigm, traceability goals, and the actual trace links [44]. As a consequence, trace link quality is often too low to be suitable for practitioners.

We found successful traceability both in plan-driven and agile cases. However, only two of the cases followed agile paradigms and had a rather rigorous culture. During the analysis, we found that culture and rigor play a critical role, in addition to the development paradigm. Had we

Table 6 Our findings and their implications

RQ	Finding	Implication
RQ1 (Sect. 4)	F1: Practitioners face several challenges related to collaboration in traceability management, including achieving traceability across organizational and tool boundaries. If not addressed, these challenges negatively impact the quality of trace links	I1: For traceability management methods and tools to be successful in practice, it is necessary to address collaborative issues and support practitioners with trace link maintenance
RQ2 (Sect. 5)	F2: Traceability has the potential of improving collaboration and supporting practitioners who communicate about dependencies of artifacts. This can give rise to new incentives to invest in traceability	I2: Both practitioners that want to improve collaboration and stakeholders investing in traceability can make use of traceability as a promising enabler if they understand the interplay of traceability and collaboration
RQ3 (Sect. 6)	F3: Traceability management in practice is generally either focused on requirements as the main artifacts and related to formal collaboration, or is driven by developers who want to trace code changes to their origin and collaborate more informally	I3: These different traceability management and collaboration approaches must be considered when investing in processes, methods, and tools
RQ4 (Sect. 7)	F4: Relevant characteristics, such as cultural and organizational aspects, are coupled with underlying development paradigms. Paradigms and the rigor of following them have a strong impact on how traceability is managed and how stakeholders collaborate. A certain level of rigor is needed to manage and make use of trace links	I4: For practitioners who want to improve their processes and use of traceability, it is recommendable to aim for consistent traceability management practices in an organization and communicate the benefit of a rigorous approach. High-quality end-to-end traceability can only be achieved if the interests of all collaborating stakeholders are supported

found these emerging aspects earlier, we could have aimed to include more agile/rigorous samples. Future research will have to scrutinize whether this finding can be generalized beyond our sample.

9 Implications of our findings

To give an easier overview of our main findings, they are presented in Table 6 together with their implications for researchers and practitioners. The findings presented in this paper allow practitioners to reflect on their use of traceability, especially with the perspective of the cost–benefit balance. This can be a starting point to establish more applicable and efficient collaborative traceability management practices. Our findings can guide tool development, processes, and the design of methods for traceability management that facilitate collaboration and establish high trace link quality. Moreover, readers interested in process-related aspects can use our findings to (re)consider traceability strategies.

We formulated the following principles:

Principles of collaborative traceability management	
P1	Put stakeholders' information needs and goals of traceability at the center
P1a	Choose a lightweight, developer-centric approach for goals related to tracking change to code
P1b	Choose a formal, requirements-centric approach for goals related to regulation (e.g., safety or legal requirements)
P2	Balance the effort and benefit of traceability management per role
P3	Enable change propagation and notification across boundaries
P4	Strive for a rigorous culture with respect to traceability maintenance

These principles are on a rather general level, and are required to be instantiated using concrete practices. Related work has listed a number of challenges and solution candidates [32]. For example, the balance of the effort and benefit of traceability can be achieved by making concerns of various stakeholders more transparent by collecting and analyzing measurable data. Our interview data gave us the impression that using a common tool for better change propagation and notification can support P3. An integrated tool was also suggested to solve traceability challenges and improve communication related to requirements [27, 32]. A rigorous culture could be enabled by a "traceability guardian" that lobbies for good trace link quality in a company, or managers ensuring that processes are followed [32].

10 Conclusion

In this paper, we presented an exploratory multiple case study on traceability management which we conducted with 24 practitioners from 15 industrial projects in Germany and Sweden.

We classified the challenges with managing traceability in a collaborative way (RQ1) as follows: (1) distributed organizational structures and tool boundaries, (2) the lack of common goals and responsibilities, and (3) collaborative trace link maintenance. Especially, the balance of cost and benefit is a challenge for practitioners, as they have to see the usefulness of investing in traceability management. Whereas this is complex in practice, we also identified positive effects of traceability management on collaboration (RQ2). If a good balance is found, both trace link quality and collaboration in an organization can be positively impacted. The challenges of traceability management and distributed software engineering do not have to exacerbate each other. Instead, traceability can support collaboration and knowledge management in distributed environments and allow the explicit documentation of decisions. It can also provide new benefits to practitioners as it can support change notification and coordination across disciplines in systems engineering.

Moreover, we analyzed how collaboration relates to traceability management approaches (RQ3): *Requirements-centered traceability management* is often related to formal collaboration procedures, as it originates in large organizations where certifications and regulations play an important role and a strict, documented development process is followed. *Developer-driven traceability management* is often conducted in less formal settings in an ad hoc fashion. It focuses especially on the implementation phase and is typically motivated by the benefit of tracking code changes back to their origin. Finally, *mixed approaches* combine the first two approaches and relate to various collaboration mechanisms. Often the requirements-centered approach is established in a top-down fashion, whereas bottom-up developer-driven approaches are introduced by the development teams.

We investigated what characteristics of the development effort have an influence on traceability management and collaboration (RQ4). These characteristics are most prominently observable in the underlying development paradigms and the rigor of following them. One finding was that traceability can be made use of independently of the level of agility. Agile approaches often leverage face-to-face communication, whereas plan-driven approaches use more formal ways of collaboration. It is, however, necessary to define traceability management strategies and follow them with a certain minimal level of rigor. If this is not the case, practitioners struggle with low trace link

quality and, consequently, with gaining benefits of their trace links.

The findings presented in this paper provide insights into the state of the practice of traceability management. Our principles of collaborative traceability management capture these findings and present them in an applicable way for practitioners.

Future work Our findings contribute toward a theory of collaborative traceability management. The presented exploratory case study which we conducted in collaboration with 24 practitioners is a starting point for future empirical research validating our findings. One idea is to conduct a quantitative study in which a survey is used to validate the identified findings and principles. We suggest to create more concrete practices based on the principles of collaborative traceability management.

One finding was that automatic approaches for traceability management are not widely used in practice while there exists a lot of research in this area. We found that practitioners are interested in using input from automatic algorithms as a suggestion, as long as the quality is satisfactory. Decision support in the form of feasible suggestions for trace links could be one way to make trace link maintenance approaches more useful. It should be kept in mind that the rationale behind each suggestion should be communicated to the user and that she or he should not be overwhelmed by the amount of suggested trace links.

Besides automatic approaches for traceability management, we found that it is an even more pressing concern to support collaboration on trace links. This can improve trace link maintenance, provided that users are not overwhelmed by messages regarding change or collaboration.

In particular, our principles need to be instantiated in concrete practices in the future. It would be interesting to explore new ways of supporting the benefits of traceability and collaboration. Based on our findings, we encourage constructive research that takes into account the cost–benefit ratio of traceability stakeholders. We see high potential for lightweight approaches that allow engineers to maintain trace link quality as a by-product of their work with low effort as well as for gamification approaches that increase intrinsic benefits.

We found that tool features supporting direct communication about trace links can be beneficial. One example is the voting feature mentioned. Future work can explore the possibility of letting practitioners comment and discuss trace links directly in a traceability management tool.

The benefit of traceability has to be conveyed to practitioners so that they perceive the benefit as outweighing the potential effort, and are thus motivated to create, maintain, and use traceability. In the future, it would be interesting to analyze usage data of trace links to identify which trace links bring added value to a company. One potential benefit can be to use tool-supported traceability as an enabler for collaboration.

Compliance with ethical standards

Funding This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation; and Vinnova [Grant Number 2014-01271 (Amalthea4public), 2014-05599 (NGEA Step 1), and 2015-04881 (NGEA Step 2)].

References

1. Ali N, Sharafl Z, Gueheneuc Y, Antoniol G (2012) An empirical study on requirements traceability using eye-tracking. In: Proceedings of the 28th IEEE international conference on software maintenance (ICSM'12), IEEE, pp 191–200. https://doi.org/10.1109/ICSM.2012.6405271
2. Arkley P, Riddle S (2005) Overcoming the traceability benefit problem. In: Proceedings of the 13th IEEE international requirements engineering conference (RE'05), pp 385–389. https://doi.org/10.1109/RE.2005.49
3. Asuncion HU, François F, Taylor RN (2007) An end-to-end industrial software traceability tool. In: Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering (ESEC/FSE'07), pp 115–124. https://doi.org/10.1145/1287624.1287642
4. Atkinson R, Flint J (2001) Accessing hidden and hard-to-reach populations: Snowball research strategies. Soc Res Update 33(1):1–4
5. Averbakh A (2014) Light-weight experience collection in distributed software engineering. Ph.D. thesis, Leibniz Universität Hannover
6. Bouillon E, Mäder P, Philippow I (2013) A survey on usage scenarios for requirements traceability in practice. In: Doerr J, Opdahl AL (eds) Requirements engineering: foundation for software quality. Springer, Berlin, pp 158–173. https://doi.org/10.1007/978-3-642-37422-7_12
7. Cleland-Huang J (2012) Traceability in agile projects. In: [8], Springer London, pp 265–275. https://doi.org/10.1007/978-1-4471-2239-5
8. Cleland-Huang J, Gotel O, Zisman A (eds) (2012) Software and systems traceability. Springer-Verlag London Limited. https://doi.org/10.1007/978-1-4471-2239-5
9. Cleland-Huang J, Gotel OCZ, Huffman Hayes J, Mäder P, Zisman A (2014) Software traceability: trends and future directions. In: Proceedings of the future of software engineering (FOSE'14), pp 55–69. https://doi.org/10.1145/2593882.2593891
10. Creswell JW (2008) Research design: qualitative, quantitative, and mixed methods approaches, 3rd edn. Sage Publications Ltd., Thousand Oaks
11. Demuth A, Kretschmer R, Egyed A, Maes D (2016) Introducing traceability and consistency checking for change impact analysis

across engineering tools in an automation solution company: an experience report. In: IEEE international conference on software maintenance and evolution (ICSME'16), pp 529–538. https://doi.org/10.1109/ICSME.2016.50
12. Easterbrook S, Singer J, Storey MA, Damian D (2008) Selecting empirical methods for software engineering research. Guide to advanced empirical software engineering, pp 285–311. https://doi.org/10.1007/978-1-84800-044-5
13. Espinoza A, Garbajosa J (2011) A study to support agile methods more effectively through traceability. Innov Syst Softw Eng 7(1):53–69. https://doi.org/10.1007/s11334-011-0144-5
14. Figueiredo MC, De Souza CR (2012) Wolf: supporting impact analysis activities in distributed software development. In: Proceedings of the 5th international workshop on cooperative and human aspects of software engineering (CHASE), pp 40–46. https://doi.org/10.1109/CHASE.2012.6223019
15. Franzago M, Ruscio DD, Malavolta I, Muccini H (2018) Collaborative model-driven software engineering: a classification framework and a research map. IEEE Trans Softw Eng forthcoming. https://doi.org/10.1109/TSE.2017.2755039. https://tinyurl.com/ya5srlvd
16. Furtado F, Zisman A (2016) Trace++: a traceability approach to support transitioning to agile software engineering. In: Proceedings of the 24th international requirements engineering conference (RE'16), pp 66–75. https://doi.org/10.1109/RE.2016.47
17. Gayer S, Herrmann A, Keuler T, Riebisch M, Antonino PO (2016) Lightweight traceability for the agile architect. Computer 49(5):64–71. https://doi.org/10.1109/MC.2016.150
18. Gotel O, Finkelstein AC (1994) An analysis of the requirements traceability problem. In: Proceedings of the 1st IEEE international conference on requirements engineering (ICRE'94), pp 94–101. https://doi.org/10.1109/ICRE.1994.292398
19. Gotel O, Cleland-Huang J, Huffman Hayes J, 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 20th IEEE international requirements engineering conference (RE'12), pp 71–80. https://doi.org/10.1109/RE.2012.6345841
20. Gotel O, Cleland-Huang J, Huffman Hayes J, ZismanA, Egyed A, Grünbacher P, Dekhtyar A, Antoniol G, Maletic J, Mäder P (2012) Traceability fundamentals. In: [8], Springer London, pp 3–22
21. Helming J, Koegel M, et al (2009) Traceability-based change awareness. In: Proceedings of the 12th intl. conf. on model driven engineering languages and systems (MODELS'09), pp 372–376. https://doi.org/10.1007/978-3-642-04425-0_28
22. Jaber K, Sharif B, Liu C (2013) A study on the effect of traceability links in software maintenance. IEEE Access 1:726–741. https://doi.org/10.1109/ACCESS.2013.2286822
23. Kannenberg A, Saiedian H (2009) Why software requirements traceability remains a challenge. J Defense Softw Eng 22(7):14–19
24. Kirova V, Kirby N, Kothari D, Childress G (2008) Effective requirements traceability: models, tools, and practices. Bell Labs Tech J 12(4):143–157. https://doi.org/10.1002/bltj
25. Klimpke L, Hildenbrand T (2009) Towards end-to-end traceability: insights and implications from five case studies. In: Proceedings of the 4th international conference on software engineering advances (ICSEA'09), IEEE, pp 465–470. https://doi.org/10.1109/ICSEA.2009.74
26. Königs SF, Beier G, Figge A, Stark R (2012) Traceability in systems engineering—review of industrial practices, state-of-the-art technologies and new research solutions. Adv Eng Inform 26(4):924–940. https://doi.org/10.1016/j.aei.2012.08.002
27. Lang M, Duggan J (2001) A tool to support collaborative software requirements management. Requir Eng 6(3):161–172. https://doi.org/10.1007/s007660170002
28. de Lucia A, Oliveto R, Tortora G (2008) IR-based traceability recovery processes: an empirical comparison of one-shot and incremental processes. In: Proceedings of the 23rd IEEE/ACM international conference on automated software engineering, IEEE Computer Society, pp 39–48. https://doi.org/10.1109/ICPC.2011.34
29. 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
30. Mäder P, Gotel O, Philippow I (2009) Motivation matters in the traceability trenches. In: Proceedings of the 17th IEEE international requirements engineering conference (RE'09), pp 143–148. https://doi.org/10.1109/RE.2009.23
31. Maro S, Anjorin A, Wohlrab R, Steghöfer JP (2016) Traceability maintenance: factors and guidelines. In: Proceedings of the 31st IEEE/ACM international conference on automated software engineering (ASE'16)
32. Maro S, Steghöfer JP, Staron M (2018) Software traceability in the automotive domain: challenges and solutions. J Syst Softw 141:85–110. https://doi.org/10.1016/j.jss.2018.03.060
33. Maxwell J (1992) Understanding and validity in qualitative research. Harv Educ Rev 62(3):279–301. https://doi.org/10.17763/haer.62.3.8323320856251826
34. Maxwell J (2012) Qualitative research design: an interactive approach. Applied Social Research Methods. SAGE Publications, Thousand Oaks
35. Myers MD, Newman M (2007) The qualitative interview in IS research: examining the craft. Inf Organ 17(1):2–26. https://doi.org/10.1016/j.infoandorg.2006.11.001
36. Palinkas LA, Horwitz SM, Green CA, Wisdom JP, Duan N, Hoagwood K (2015) Purposeful sampling for qualitative data collection and analysis in mixed method implementation research. Adm Policy Ment Health 42(5):533–544. https://doi.org/10.1007/s10488-013-0528-y
37. Panis MC (2010) Successful deployment of requirements traceability in a commercial engineering organization...really. In: Proceedings of the 18th IEEE international requirements engineering conference (RE'10), pp 303–307. https://doi.org/10.1109/RE.2010.43
38. Rahimi M, Cleland-Huang J (2018) Evolving software trace links between requirements and source code. Empir Softw Eng 23(4):2198–2231. https://doi.org/10.1007/s10664-017-9561-x
39. Ramesh B (1998) Factors influencing requirements traceability practice. Commun ACM 41(12):37–44. https://doi.org/10.1145/290133.290147
40. Ramesh B (2002) Process knowledge management with traceability. IEEE Softw 19(3):50–52. https://doi.org/10.1109/MS.2002.1003454
41. Ramesh B, Stubbs C, Powers T, Edwards M (1997) Requirements traceability: theory and practice. Ann Softw Eng 3(1):397–415. https://doi.org/10.1023/A:1018969401055
42. Rath M, Rendall J, Guo JLC, Cleland-Huang J, Mäder P (2018) Traceability in the wild: automatically augmenting incomplete trace links. In: Proceedings of the 40th international conference on software engineering, ACM, New York, NY, USA, ICSE '18, pp 834–845. https://doi.org/10.1145/3180155.3180207
43. Rempel P, Mäder P (2015) A quality model for the systematic assessment of requirements traceability. In: Proceedings of the 23rd IEEE international requirements engineering conference (RE'15), pp 176–185. https://doi.org/10.1109/RE.2015.7320420
44. Rempel P, Mäder P, Kuschke T (2013) An empirical study on project-specific traceability strategies. In: Proceedings of the 21st IEEE international requirements engineering conference (RE'13), pp 195–204. https://doi.org/10.1109/RE.2013.6636719

45. Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng pp 131–164. https://doi.org/10.1007/s10664-008-9102-8
46. Runeson P, Höst M, Rainer A, Regnell B (2012) Case study research in Softw Eng. Wiley, Hoboken
47. Santiago I, Jiménez Á, Vara JM, De Castro V, Bollati VA, Marcos E (2012) Model-driven engineering as a new landscape for traceability management: a systematic literature review. Inf Softw Technol 54:1340–1356. https://doi.org/10.1016/j.infsof.2012.07.008
48. Schneider K (2006) Rationale as a by-product. In: Dutoit AH, McCall R, Mistrík I, Paech B (eds) Rationale management in software engineering. Springer, pp 91–109. https://doi.org/10.1007/978-3-540-30998-7_4
49. Schneider K (2009) Experience and knowledge management in software engineering. Springer, Berlin. https://doi.org/10.1007/978-3-540-95880-2
50. Seaman CB (1999) Qualitative methods in empirical studies of software engineering. IEEE Trans Softw Eng 25(4):557–572. https://doi.org/10.1109/32.799955
51. Sekitoleko N, Evbota F, Knauss E, Sandberg A, Chaudron M, Olsson HH (2014) Technical dependency challenges in large-scale agile software development. In: Cantone G, Marchesi M (eds) Proceedings of international conf. on agile softw. dev. (XP'14), Springer, Rome, Italy, LNBIP, vol 179, pp 46–61. https://doi.org/10.1007/978-3-319-06862-6_4
52. Sengupta B, Chandra S, et al (2006) A research agenda for distributed software development. In: Proceedings of the 28th international conference on software engineering (ICSE'06), pp 731–740. https://doi.org/10.1145/1134285.1134402
53. Sengupta S, Kanjilal A, Bhattacharya S (2008) Requirement traceability in software development process: an empirical approach. In: Proceedings of the 19th IEEE/IFIP international symposium on rapid system prototyping (RSP'08), IEEE, pp 105–111. https://doi.org/10.1109/RSP.2008.14
54. Sinha V, Sengupta B, Chandra S (2006) Enabling collaboration in distributed requirements management. IEEE Softw 23(5):52–61. https://doi.org/10.1109/MS.2006.123
55. Spanoudakis G, Zisman A (2005) Software traceability: a roadmap. In: Handbook of software engineering and knowledge engineering, vol 3. World Scientific Publishing, pp 395–428
56. Strašunskas D (2002) Traceability in collaborative systems development from lifecycle perspective. In: Proceedings of the 1st international workshop on traceability in emerging forms of software engineering (TEFSE '02), pp 54–60
57. Sundaram SK, Hayes Huffman J, Dekhtyar A, Holbrook EA (2010) Assessing traceability of software engineering artifacts. In: Proceedings of the 18th IEEE international requirements engineering conference (RE'10), pp 313–335. https://doi.org/10.1007/s00766-009-0096-6
58. Wohlrab R, Steghöfer JP, Knauss E, Maro S, Anjorin A (2016) Collaborative traceability management: challenges and opportunities. In: Proceedings of the 24th IEEE international requirements engineering conference (RE'16), pp 216–225. https://doi.org/10.1109/RE.2016.17