FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Department Informatik

Technical Reports / ISSN 2191-5008

Julia Krause, Andreas Kaufmann, Dirk Riehle

The Code System of a Systematic Literature Review on Pre-Requirements Specification Traceability

Technical Report CS-2020-02

August 2020

Please cite as:
Julia Krause, Andreas Kaufmann, Dirk Riehle , “The Code System of a Systematic Literature Review on Pre-Requirements Specification Traceability,” Friedrich-Alexander-Universität Erlangen-Nürnberg, Dept. of Computer Science, Technical Reports, CS-2020-02, August 2020.

Friedrich-Alexander-Universität Erlangen-Nürnberg
Department Informatik

Martensstr. 3 · 91058 Erlangen · Germany

www.cs.fau.de

The Code System of a Systematic Literature Review on Pre-Requirements Specification Traceability

Julia Krause, Andreas Kaufmann, Dirk Riehle Professorship for Open Source Software
Dept. of Computer Science, University of Erlangen, Germany
julia.krause@fau.de, andreas.kaufmann@fau.de, dirk@riehle.org

Abstract

Tracing requirements back to their origin is important to understand the context and to identify all potential concerns or conflicts in the genesis of the requirements. The so-called pre-requirements specification (pre-RS) traceability is part of requirements traceability (RT) and is recognized as an important factor for long-term project success. Different approaches have been developed to connect requirements and their origin, but still, there is no general established solution. Mostly it is seen as an effort without significant benefits leading to resistance towards its implementation. To guide future research, a comprehensive understanding of the use cases, benefits, challenges, and existing approaches encountered is vital. However, the landscape of the scientific literature on this field remains somewhat fragmented, with pre-RS traceability often only considered in the periphery. We, therefore, conducted a systematic literature review, which identified 67 relevant papers. The literature was coded using qualitative data analysis (QDA) methods. In this technical report, we present the whole resulting code system of the QDA and describe the relationships between the identified themes, focusing on the relevance, problems, solutions, and techniques of pre-RS traceability.

Index Terms

Pre-Requirements Specification Traceability, Requirements Traceability, Requirements Engineering, Qualitative Data Analysis

Contents

1 Introduction

What is the reason behind a requirement? Why was this requirement changed and who was or is involved? These questions are becoming more important especially for long-term or continuously changing systems. Pre-requirements specification (pre-RS) traceability can answer these questions by providing trace links between requirements and their origins. On the other side, post-RS traceability links requirements and artifacts based on the requirements specification (RS). Compared to post-RS traceability, research on pre-RS traceability is sparse. As early as in 1992, Gotel stated that "[pre]-RS traceability has the potential for much greater leverage than post-RS traceability on costs and quality of software produced" [1]. Yet, to this date, the space of the academic literature on pre-RS traceability has not been systematically laid out.

Pre-RS traceability is a remarkably dynamic procedure. It handles a lot of unstructured, qualitative, continuously changing, and varying data, like meeting protocols, change requests, etc. Especially, at the beginning of each project many decisions, discussions, and negotiations are made yielding much information that must be considered in the RS. Past decisions should be preserved to assist in future decision-making. Thus, repeating wrong decisions can be avoided [2], [3].

Understanding all involved persons, their needs, and their different roles in the project become crucial [4]–[7]. Additionally, finding an appropriate traceability technique or tool is important. Each project setting is unique, therefore generic tools have to be customized or individual tools have to be developed [8]. However, the effort of establishing and maintaining pre-RS traceability is often seen as too high [9]–[11]. So it is commonly only applied to fulfill regulatory compliance for safety-critical systems, without realizing the full potential of these traces.

Keeping track of the origin of a requirement provides many benefits like presenting the state of the RS by proving fulfillment of stakeholder needs [10], [12]. Pre-RS traceability supports communication because involved persons can be identified and contacted [4], [13]. Furthermore, analysts or requirements engineers feel much more prepared for negotiations, knowing details about the requirements creation and decisions behind a particular requirement [12], [14].

Nevertheless, pre-RS traceability is commonly handled as a sub-topic of requirements traceability (RT) which is still a big challenge. In this work, we want to focus on pre-RS traceability to document the state-of-the-art by analyzing current issues, scenarios, techniques, and tools. We conducted a systematic literature review (SLR) of articles published in 13 academic journals and over 15 conference proceedings from January 1992 until June 2020. As part of the review process we performed qualitative data analysis (QDA) on the 67 papers we retrieved as relevant to our topic. The focus of this technical report is based on the resulting code system of our QDA to provide an overview of the different topics and their prevalence in published literature. The main purpose of this report is to serve scientists who are interested in a specific area of pre-RS traceability. Additionally, we are working on a SLR paper including findings based on the code system and answers to our three guiding research questions.

This technical report is structured as follows. First, section 2 provides a brief overview of our research method including the QDA from which the presented code system results. The code system is then presented in detail in section 3. Section 4 discusses the code system and related finding. We close the report by drawing a conclusion in section 5.

2 Research Method

We set out to map the field of pre-RS research in three main dimensions: When is it used? What problems does it solve? And how does it solve the problem? To guide our analysis of the published literature, we refined these dimensions of interest into the following three research questions:

RQ1) Relevance - What are the use cases and benefits of pre-RS traceability?

RQ2) Problems & Solutions - What are the problems and solution approaches of pre-RS traceability?

RQ3) Techniques - What are pre-RS traceability techniques?

Guided by these research questions we performed a pre-structured (deductive) qualitative literature review following the process of Kitchenham [15]. To plan and document our complete audit trail we created a **research protocol** that includes background information and important cornerstones like research questions, search strategy, selection and quality criteria, research process, the data extraction process, and work program.

We retrieved literature from well-known **scientific databases** (Google Scholar, IEEE Xplorer, ACM Digital Library, Web of Science, and Springer Link) by keyword searching. Our basic search term was "pre-requirements specification traceability" which we varied in the different spellings and abbreviations we found. The keyword search resulted in 36 relevant papers. Based on these 36 papers we carried out a forward and backward search. During this search, we uncovered 31 additional papers about pre-RS traceability because some of the relevant papers do not contain any varying terms of pre-RS traceability.

To identify literature as relevant we developed qualitative and topic-specific criteria. We analyzed a pilot sample of ten papers to test and refined the criteria. As **qualitative criteria**, the literature has to be written in English and have to be peer-reviewed. We do not exclude non-evaluated techniques, because we would lose interesting ideas and well described theoretical approaches. Our **topic-specific inclusion criteria** were:

- Literature about a technique or method to link origin artifacts with RS
- Literature about an overview which presents different techniques, issues and/or problems to link origin artifacts with RS or issues related to this topic

- Literature about an evaluation of a technique or method which links origin artifacts with RS

We refined the criteria to exclude literature that only mentioned pre-RS traceability to be given completeness and not going into details of this topic.

To analyze the relevant literature we conducted a **QDA** tool-supported by MAXQDA[1]. The iterative coding process consists of the three steps: *open coding*, *axial coding* and *selective coding* [16]. During our QDA we performed peer debriefings to continuously check and improve the quality of our research [17]. The following section 3 explains the whole code system and their codes into more detail.

3 Code System

The main codes of the code system are represented in table 1 including their number of codings in the right column. Not all main codes are directly related to the research questions. They were created during the QDA to capture additional important phenomenon and background information.

The following subsections present the child codes of each main code. Each code is described by a summary of the findings based on the referenced literature in the right column. The penultimate column contains the number of codings or marks *group-code-only (GC)*. A GC does not contain any codings it just groups child codes by similarity. Mentioned relationships between codes within the short summaries based on the code-relations-analysis of MAXQDA. The code-relations-analysis identifies two codes as related if the coded text segments overlap each other.

During data analysis, we aimed to focus exclusively on information specific to pre-RS traceability and avoid general RT or post-RS traceability related information. Due to the strong correlation between pre-RS traceability and post-RS traceability as subordinate topics of RT, this was not always possible. Therefore, we made sure to include content in the code system which is only or also related to pre-RS traceability.

TABLE 1: Main codes of the code system and their number (No.) of codings includes

Code and description	No. of codings
Requirements Traceability (RT)	5
Pre-RS traceability general (+ sub codes)	94
Traceability users (+ sub codes)	31
Use cases & Benefits (+ sub codes)	220
Problems & Challenges (+ sub codes)	242
Solutions & Suggestions (+ sub codes)	249
Traceability tool (+ sub codes)	47
Traceability models/techniques (+ sub codes)	193
Consequences of poor pre-RS traceability	4

3.1 Requirements traceability & pre-RS traceability general

The first part of the code system presented in table 2 is not directly related to the research questions. However, the main codes *requirements traceability (RT)* and *pre-RS traceability general* provide basic information to gain a common ground for further elaborations. The child codes of *pre-RS traceability general* are ordered by descending number of codings.

Analyzing the pre-RS traceability definitions and descriptions, we discovered conflicting views. As the codes below show, some literature excludes inter-requirements traceability as part of pre-RS traceability. *Inter-requirements traceability* consists of trace links between two related requirements or two versions of one requirement. Other literature included inter-requirements traceability as part of pre-RS traceability. We created two codes to model this contradiction. Comparing the number of codings of these two codes *definition/descriptions > exclude inter-requirements traceability* and *definition/descriptions > include inter-requirements traceability* more literature includes inter-requirements traceability. Based on this and our experience gained from the analysis we also refer to inter-requirements traceability as part of pre-RS traceability. For example in the case of refinements, the origin of a newer requirement can be the previous version and a particular decision. This would lead to two pre-RS trace links.

TABLE 2: Child codes of the codes *requirements traceability (RT)* and *pre-RS traceability general*

Requirements traceability (RT): This code collects some descriptions about RT. RT ensures that the requirements satisfy the stakeholder needs by linking them to the origin and to further design or development artifacts. RT is part of the requirements management and is seen as a tool to handle complexity by uncovering dependencies between requirements and other artifacts.	5	[7], [18]–[21]

1. https://www.maxqda.com

Pre-RS traceability general: The collection of child codes combine information about what pre-RS traceability is, what are their metrics and different traceability types.	GC	
Definition/Descriptions: According to the literature the definition of pre-RS traceability and its differentiation from other terms is clear. Pre-RS traceability "[...] refers to those aspects of a requirements's life prior to inclusion in the RS." [9]. Most of the literature refers to this definition. Sometimes it is also called upstream tracing [22].	17	[1]–[3], [6], [9], [10], [20], [23]–[29]
Include inter-requirements traceability: Some papers are explicit in including requirements refinements and requirements-to-requirements relationships in pre-RS traceability.	4	[4], [6], [11], [30]
Exclude inter-requirements traceability: Some papers exclude requirements-to-requirements relationships from pre-RS traceability.	2	[20], [31]
Types of requirements origin artifacts: Connecting requirements with their origin requires to take a closer look on the different characteristics of this origin artifacts. They can be media objects like videos, images or recordings or they can be document based. Documents can contain unstructured, (semi-)formal natural language or formal content.	19	[1], [10]–[12], [18], [20], [21], [32]–[35]
Delimitation of Types: This code contains overviews and comparisons of different requirements traceability types with focus of pre-RS traceability. The different types can overlap thematically and are represented by the child codes.	13	[1]–[3], [6], [9], [10], [26], [30], [36]
Inter-requirements traceability: Tracing between requirements is necessary to understand their dependencies between different requirements. One use case is the impact analysis. Furthermore, it also describes the tracing between different requirements versions created during requirements evolution.	6	[6], [20], [25]
Pre-RS forwards traceability: The content of this code describes the tracing from an origin artifact to one or more requirements.	5	[1], [11], [19], [20]
Pre-RS backwards traceability: The content of this code describes the tracing from a requirement back to the origin artifact(s).	5	[1], [11], [19], [20]
Requirement-rationale: The rational provides information about why the requirement exists.	3	[20], [31]
Requirement-stakeholder/roles: This link traces from a requirement to the owner or other involved persons.	3	[20], [31]
Multiplicity between origin & target: This code is about the direction of the link. The trace link can be unidirectional or bidirectional without effecting the navigation between artifacts.	2	[3]
Upstream: Upstream is associated with pre-RS traceability, going backward from a requirement to its origin. The opposite is downstream tracing related to post-RS traceability.	2	[22]
Vertical traceability: This includes tracing between elements of the same model, for example tracing between different abstraction levels.	2	[26], [31]
Horizontal traceability: The opposite of vertical traceability describes the tracing between elements of different models or versions, like tracing between different requirement groups or to other artifacts.	2	[26], [31]
Extra-requirements traceability: It describes all non inter-requirements relationships, means all trace links to other artifacts. In the case of pre-RS traceability, extra-requirement traceability describe links between requirements and their input artifacts.	1	[6]
Functional tracing: Tracing between the functional aspects of the software system is called functional tracing.	1	[6]
Non-functional tracing: It describes all traces which involved non-functional requirements.	1	[6]

Traceability activities: To create and manage pre-RS traceability different activities are necessary: (1) planning for traceability, (2) recording/extracting of traces: this can be done immediately during the writing of related requirements or by recovering methods. A recovering method typically contains the steps parsing documents, generating and afterward evaluating the candidate links. (3) Using traces by performing traceability analysis or generating of reports and (4) maintaining traces.	7	[3], [37]–[40]

3.2 Traceability users

A large part of the literature mentioned traceability users and their roles as a highly influencing factor in RT. Therefore, it is necessary to dive into details of different user types, their roles, and activities for pre-RS traceability. We observe that the distinction of different types is mainly based on traceability practice and motivation or the traceability function and capabilities. The user categories of Ramesh (high-end and low-end users) [41] and Gotel and Finkelstein (provider and end-users) [9] are often referenced.

Table 3 presents the code *traceability users* including references and more specific child codes. The codes of the table are ordered by descending number of codings.

TABLE 3: Child codes of the *traceability users* code

Traceability users: This code contains overviews and comparisons of different traceability user types. This user types are represented by the child codes.	4	[42]–[44]
User types by motivation/practice: Different types of traceability users were recognized. The types high-end and low-end users distinguished by traceability practice are often referenced by other literature. Following types are also distinguished by motivation and practice [43]: (1) the regulated, (2) the sub-contractor, (3) the consultant and (4) the enthusiast	4	[41], [43], [45]
High-end users: "High-end users view traceability as an important component of a quality system engineering process". They know how to use traceability and its benefits. Studies uncovered that this knowledge is based on more years of experience than low-end users have.	10	[41]
Low-end users: They see traceability as mandatory and regulation by stakeholder. They only do what is necessary and see no further benefits.	8	[41]
Provider: Only they can create trace links. Therefore, providers spent the time and effort for creation and maintenance, but there is a danger that they often don't know the need and benefit of a link. That leads to misunderstandings and unfitting or incomplete trace links.	3	[9], [37]
End-user: Typical end-users have to work with traceability. Different end-users have different interests and requirements for trace links, but they are typically not able to create these links themselves.	3	[1], [9], [46]

3.3 Use cases, benefits & consequences

The answers to research question RQ1 are presented in table 4. RQ1 itself and their answers refer to the relevance of pre-RS traceability. During the QDA we added the code *consequences of poor pre-RS traceability*. Not many codings could be found for this code, but it contributes to the motivation of establishing pre-RS traceability.

The codes of table 4 are ordered by descending number of codings. The most commonly stated use case for pre-RS traceability is the identification of the source and the responsibilities or the fulfillment of regulations for safety-critical systems. The prevalence of these use cases is reflected in the number of codings for this code. But beyond that, the literature refers to a wide variety of 14 different use cases and nine benefits.

TABLE 4: Child codes of the *use cases* and *benefits* codes

Use cases: This code just groups all identified use cases.	GC	
Find source (support understanding): Finding the source of a requirement is the most mentioned reason for pre-RS traceability. This use case describes the tracing between requirements and other origin artifacts except personal contacts (coded by the following code). Finding the source goes together with the codes *responsibility identification* and *track history/relationships*. The benefits are to monitor & gain knowledge for future, improving communication & collaboration and supporting reusability of requirements and trace information.	36	[3], [4], [6], [9]–[12], [19]–[21], [28], [33], [38], [42], [43], [47]–[51]
Responsibility identification: This code is strongly related to the code above *find source (support understanding)*. But unlike the code above, responsibility identification describes one or more added contacts to a requirement. A contact includes specific contact information like mail address, telephone number, and role information.	18	[4], [6], [9]–[12], [19], [20], [38], [41], [47], [48], [52], [53]
Fulfillment of regulatory/compliance/norms: This use case is related to RT, including pre-RS traceability. Especial for security or safety-critical systems, it becomes necessary to prove the fulfillment standards. Therefore, pre-RS traceability can present the reason and the regulation of particular requirements. Related benefits are the reduction of maintenance costs, monitor & gain knowledge for the future, and the improvement of the product/software quality.	17	[7], [10]–[12], [23], [30], [37], [43], [54]
Prove fulfillment of stakeholder needs: This code is strongly related to the code *fulfillment of regulatory/compliance/norms*. Especial for pre-RS traceability, it proves the right to exist of each requirement based on the stakeholder needs. Furthermore, it is possible to uncover needs that are not satisfied by the RS.	10	[3], [10], [12], [20], [37], [42]
Coverage analysis: This analysis is a valuable tool general for RT. The coverage analysis allows an impression of the status of the RS. Enthusiastic traceability users see it as important, but with less priority.	4	[12], [43]
Get the state of the RS: Knowing the state of an RS is a necessary part of . the overall software project process tracking. Questions like, "Which stakeholder needs are satisfied by requirements?" and "How many stakeholder needs are not considered inside the RS?" can be answered. This code is related to the *coverage analysis*	10	[3], [10]–[12], [24], [41], [43]
Change-Management: This code represents an important use case for RT. Pre-RS traceability support the integration of a new decision into the RS by analyzing the impact and localizing necessary updates. The following child codes provide more specific use cases. Change-Management is related to the codes *requirements prioritization* and *finding the source (support understanding)*. The benefits are satisfaction of stakeholder, supporting reusability of requirements and improving product/software quality.	8	[3], [12], [20], [24], [46]
Impact Analysis: Performing an impact analysis is a general RT use case. It is especially used in lager projects where many different persons and roles are involved. Existing pre-RS trace links improves the impact analysis.	15	[3], [7], [10], [11], [19], [20], [31], [41], [42], [47], [48]
Capture & support decisions: This code is pre-RS traceability specific. Capture decisions and their results support negotiations and future decision-making. Repeating errors can be prevented for example by analyzing decisions in the past.	14	[2], [3], [7], [9], [12], [13], [36], [47], [49], [55], [56]
Track history/relationships: Tracking the history and relationships of requirements becomes more important the bigger a project is. This code is strongly related to the code *find source (support understanding)* and has two more relationships to the codes *requirements negotiations* and *management of system evolution (maintaining)*. This tracking supports the gathering of knowledge for the future, reusability, communication, and collaboration within the team and to the stakeholder.	9	[11], [12], [28], [48], [49]

Manage system evolution (maintaining):Nearly every project has to handle the system evolution. RT including pre-RS traceability provides valuable support for maintenance. Obviously, this code is related to the codes *track history/relationships* and *finding the source (support understanding)*. The benefits are to monitor & gain knowledge for the future, supporting reusability, improving communication & collaboration, reducing maintenance costs, and improving product/software quality.	10	[5], [10], [12], [27], [35], [51]
Knowledge management system: Collecting knowledge and experiences by a knowledge management system supports planning, avoids later mistakes and conflicts, and supports the reusability of requirements.	7	[12], [35]
Requirements negotiations: Pre-RS traceability is valuable for negotiations because it provides comprehensive background information about requirements creation and knowledge to find arguments and to discover potential conflicts and solutions. This code is related to the use case codes *track history/relationships*, *capture & support decisions*, *find source (support understanding)*, and *manage system evolution (maintaining)*. The benefits are to monitor & gain knowledge for the future, support reusability of requirements, and improve communication & collaboration.	5	[12], [14]
Requirements prioritization: Knowing the context and rationals behind requirements supports requirements prioritization. This code is related to the codes *change-management*, *proving fulfillment of stakeholder needs*, and *find source (support understanding)*. The benefit is reusability of requirements.	5	[3], [20]
Benefits: This code just groups all identified benefits.	GC	
Monitor & gain knowledge for future: This code describes a general benefit of RT. The monitoring of the pre-RS traces provides an overview of the status of the RS. Especially high-end users said it is important to gain knowledge for future improvements.	16	[3], [12], [28], [41], [49]
Improve product/software quality: This code describes a general benefit of RT. Keeping the RS updated and on high quality is essential for all artifacts based on it, especially if the system becomes more complex. Reduction of maintenance costs, reusability of requirements, and improvement of product/software quality are related to each other.	10	[3], [5], [10], [12], [28], [35], [54], [56]
Support reusability of requirements and traces: Reuseability is a benefit of RT including pre-RS traceability, by saving time and cost. This code is related to the other benefits monitoring & gaining knowledge and improving communication & collaboration.	10	[3], [10], [12], [20], [41]
Improve communication & collaboration: Pre-RS trace links clarify responsibilities and provide transparent documentation. This improves communication within the team and to the stakeholders, especially in distributed work environments. The code is related to monitoring and reusability of requirements and trace information.	7	[4], [12], [13], [35]
Reveal tacit knowledge: On the one hand, tacit knowledge is the knowledge that is only in the mind of a stakeholder and remains unspoken. On the other hand, it can be knowledge of the requirements engineer that flows into the RS. Some literature uncovered that requirements, without traces to their origin artifacts, may be based on such tacit knowledge.	4	[2], [57], [58]
Satisfaction of stakeholder: An option to satisfy stakeholders is to provide them the opportunity to follow the project progress. In the case of pre-RS traceability, the stakeholder can understand and see which needs are satisfied by which requirements.	2	[11], [20]
Finding missing requirements: According to the above benefit *satisfaction of stakeholder* it is possible to uncover stakeholder needs not included in the RS.	1	[10]
Reduction of maintenance costs: In the case of rewriting and refinement of requirements, pre-RS traceability provides additional background information, refinement history, or persons in charge.	1	[10]

Finding unnecessary requirements: Pre-RS traceability uncover requirements that do not have a trace back to their origin. These requirements need further observations and if they are not based on tacit knowledge they may be unnecessary requirements.	1	[10]
Consequences of poor pre-RS traceability: Studies uncovered that the main reason for poor RT is a lack of pre-RS traceability even before post-RS traceability, especially for long-term projects. Missing documentation about the building process or the context leads to "back box" requirements. "Black box" requirements prevent adequate requirements evolution.	4	[4], [9], [21], [24]

3.4 Problems & Challenges

One part of the answer to RQ2 about current problems and challenges in pre-RS traceability is presented in table 5. To structure this part of the code system all codes and their child codes are ordered by descending number of codings. That identifies the *person-related problems* including all child codes as the most frequently used code under the *problems & challenges* code. According to the literature, this also proves that the human factor is one of the most critical points in pre-RS traceability. Different roles and different interests are always involved in a project. Therefore, providing an appropriate pre-RS traceability strategy supporting all the different interests is a big challenge. *Different interests of different roles* is the most coded person-related problem followed by the known problem code *too much work for unseen benefits*.

TABLE 5: Child codes of the *problems & challenges* code

Problems & Challenges: This code contains overviews of problems. The child codes represent special problems and challenges.	4	[3], [8], [41], [49]
Person-related problems: Much research was done in the area of tools and techniques to achieve traceability, but not so much about the influence of the human factor and their social aspects. The human factor is one important part of the multifaceted RT problem. The child codes summarize the most frequently mentioned problems.	4	[8], [21], [59]
Different interests of different roles: The involvement of different project members or parties and their individual knowledge and viewpoints leads to varying traceability usage, interest, and need. An often mentioned conflict exists between providers and end-users because end-users can not predefine every traceability need beforehand. The challenge is to satisfy both parties at the beginning and during the project period	23	[1], [3], [8], [9], [18], [24], [26], [40], [41], [44]–[46], [51], [59]
Too much work for unseen benefits: Too much effort for too few benefits is a significant problem in establishing pre-RS traceability. The benefits are often not clearly explained to all involved especially, for the providers. This additional time effort impacts the individual productivity. "if nobody pays you to document trace, then you don't do it" is the argument of low-end users [41]. Only superficially implemented traceability is hard or not possible to maintain, too. The causes of this problem are *no/few or bad (semi-)automation* of trace links and *missing trace usage goals*.	18	[1], [3], [9]–[12], [24], [26], [37], [41], [60], [61]
No trust in traces: No trust in trace links is caused among others by *no/few or bad (semi-)automation*, a *lack of verifying correctness & completeness*, and *no/poor maintenance of trace information*. Finally, the time wast of following a wrong trace link down the system and not satisfied trace expectations leads to no trust in traces and the used tool.	7	[3], [6], [8], [10], [13], [26], [37]
Responsibility problem: Ad-hoc effort, no guidelines, bad automation, or too much work causes a responsibility problem to find the person in charge of general traceability topics or particular requirements. Additionally, the larger the project becomes, the more people are involved. Without a responsible person, the maintenance of the links is also at risk.	7	[1], [8], [21], [36], [52]

No immediate benefit seen: No immediate benefits demotivate all users, but especially trace providers. The benefits do not occur where the effort (cost) is. In the field of post-RS traceability, there was much done for visualization of trace information to improve accessibility. But still, there is a lack of visualizations for pre-RS traceability. This code is related to the problem of *low priority* and *bad information access/presentation*.	3	[11], [12], [47]
Subjective/idealized traces: Compared to the code *different interests/knowledge of different roles*, this code distinguishes between the understanding of different individuals their subjective and idealized assessment of traces. The challenge is to build a common, objective knowledge base.	2	[1], [51]
Inadequate documentation of trace information: Losing requirement production information leads to "black box" requirements. Later it is nearly impossible to retrace the production process. Missing information leads to incomplete trace documentation then the documentation becomes unstructured and unmanaged and hard to maintain. Missing trace information harbors the dangers of excusing a product that does not satisfy the stakeholder.	6	[5], [14], [21], [33], [62]
Tacit knowledge/unnoticed links: Tacit knowledge is a common phenomenon during information gathering and RS creation. It describes knowledge that is only in mind, but no one speaks about it. A stakeholder didn't speak about, and an analyst includes their tacit knowledge into the RS. Tacit knowledge makes pre-RS traceability difficult. But a pre-RS traceability analysis potentially uncovers such tacit knowledge. Another related problem is unnoticed links often due to an insufficient presentation of trace information.	11	[2], [3], [9], [37], [46], [57], [58]
Ad-hoc effort: Establishing traceability by individual motivated persons, only locally on some project parts, or at the beginning of the project until the first version of the RS finished, is a common phenomenon. No responsible person is defined, and the workload is too high for continuous traceability. Often the ad-hoc effort is to satisfy the stakeholder or to achieve required audit results. Studies observed that the argument for no maintenance is that all traceability benefits are already achieved.	8	[1], [9], [10], [46], [54]
Lack of verifying correctness & completeness: Until now, there exist no mechanisms for verifying completeness and correctness of trace information to establish high trace quality. This can lead to loss of trust. Especially for (semi-)automated trace generation approaches control mechanisms are essential for trustworthy trace links.	3	[7], [26]
No documentation of verbal communication/interaction: A significant aspect that leads to incomplete requirements sources lies in not recording/documenting verbal communications. Especially in agile projects where discussions about requirements are an essential part.	5	[1], [3], [5], [21]
No/poor maintenance of trace information (during evaluation): If a project becomes more complex and more durable, it gets more difficult to keep all living traceability documents up to date. Only a few outdated trace links can cause many time-consuming problems. A study uncovered that only seven out of ten analysts maintain trace information [12]. So the challenge is to handle this information and to establish continuous maintenance to avoid "back-boxed" requirements, time-consuming maintenance, and to support validation and verification of requirements.	15	[8]–[10], [12], [19], [21], [24], [36], [41], [46], [61]
Path ephemerality problem: During software evolution, it gets difficult to keep all trace links up to date, especially for long-term projects. Often this is caused by a lack of change culture and inadequate support by tools or organization. The information has been lost or deleted. For example, an existing trace link pointed to a non-reachable target.	10	[1], [3], [5], [8], [9], [21], [41], [54]

No generalizations/standards: The lack of guidelines, best practice, and standards (company-intern or general) leads to more individual effort. For example, the creation and improvement of best practices for adapting traceability strategies need active feedback loops. High-end users recognized the absence of guidelines and well-defined methodologies as a problem. The lack of standards leads to inadequate or no exchange of traceability documents between different environments. An adequate set of guidelines and standards can guide the traceability process, including all necessary artifacts.	22	[1], [3], [8], [9], [21], [26], [28], [34], [38], [41], [43], [45], [56], [63]
Organizational problems: The lack of commitment and support by the company are organizational problems. Among others, this leads to no or poor technology transfers and knowledge sharing between different companies or projects within a company. Studies pointed out that the problems of traceability are more organizational than technical.	5	[1], [6], [8], [59]
Collaboration across boundaries: The exchange of traceability information across internal or external organizational boundaries is difficult, if not impossible, due to different environments (different tools, languages, terminologies). This leads to maintenance problems. Studies revealed that the problem of collaboration across boundaries was caused by a lack of communication.	8	[8], [13], [41], [43], [61]
Low priority of traceability: Post-RS traceability has a higher priority as pre-RS traceability. Often, pre-RS traceability is seen as optional extra or side activity. Therefore, not enough resources are available to establish and maintain trace information. This leads to no or poor adaptation of traceability to project-specific needs.	7	[1], [7], [9], [11], [26], [36]
Tooling problems: Tool support is necessary to handle the amount of traceability data. But still, a lack of adequate tool support exists. Different tools were developed, but many of them only address limited parts of the development life cycle or do not support all types of artifacts. Therefore, customizable tools are necessary. Companies often complain about high maintenance and adoption costs for traceability tools.	8	[3], [4], [8], [11], [26], [41], [43]
Exchange between tools: A software project usually consists of many tools to perform different project tasks. Communication between different traceability tools is essential for the exchange of traceability information. Without standards, however, it will be difficult, if not impossible, to find a common exchange format.	3	[3], [26], [43]
Limited support for interaction/collaboration: Existing tools can not fully track collaboration and interaction during information gathering or later discussions. Especially to handle information from many stakeholders, it is necessary to keep an overview. However, without tracking collaboration and interaction important information got lost. This leads to unclear pre-RS traceability and incomplete documentation.	2	[62]
Bad access/presentation of trace information: Valuable visualizations and presentations of pre-RS traceability information are noticeable rare. One challenge is to predefine all visualization types fitting for different use cases. For pre-RS traceability, it is an often referenced problem to access the sources of requirements. An adequate visualization can process a large amount of trace information, avoids problems with unnoticed links, and provides immediate benefits.	11	[1], [3], [9], [12], [33], [37], [43]
Missing usage goals for traces: The identification of a clear trace link usage goal is necessary to drive the trace activities. If usage goals of trace links are missing, no one will understand the need for the trace links. Studies identified missing trace link usage goals in many cases or the mismatch between reported trace link goals and the current existing trace links.	5	[3], [54]
Trace Path suitability problem: The usage of trace links differs depending on activity and user. If the trace link did not fit the use case, it can only be queried selective.	3	[51], [54]

No/poor identification of trace artifacts: Establishing and using a pre-RS traceability strategy needs previous definitions of trace links and their artifacts, to know what is relevant to trace. Therefore, a trace link has to be registered to get more information, to prove their existence, and to use it. A non-registered link can not be found. The identification of correct trace links and trace artifacts also needs a definition of trace information granularity. Choosing the right granularity is challenging because wrong granularity leads to over-complexity or little information. Another challenge is to avoid ambiguity trace artifacts by using a unique identifier.	8	[6], [7], [43], [46], [54]
No/poor connection of different object types: The characteristic of an origin artifact can be of different types like text documents or media objects. The challenge for pre-RS traceability is to support trace links that can trace between these different artifact types and the requirements.	2	[45], [49]
Organize unstructured information: Especially pre-RS traceability have to connect different widely varying artifacts with the RS. These artifacts are mostly different types of unstructured documents (in natural language or formal) or media objects. They must be stored in such a way that traceability is guaranteed. Managing a large amount of source data is still a challenge.	8	[1], [3], [6], [9], [12], [46], [61]
No/few or bad (semi-)automation: A wide variety of different automation or semi-automation approaches exist to generate or recover trace links. But still, no solution does not create erroneous trace links. Combined with the lack of mechanisms to verify correctness and completeness, the manual effort to check the candidate links is high. Keeping wrong links in a trace set can lead to loss of user trust.	9	[3], [8], [24], [26], [37], [64]
No/few support managing of large/distributed amount of trace data: Many case studies have only been done with small data sets or small projects. Therefore, the knowledge about pre-RS traceability techniques working for large and scaled projects is rare. Mostly pre-RS traceability is done manually and becomes too time-consuming or impossible in real projects.	5	[2], [7], [37], [56], [61]
Poor adaptability for project specific needs: Different projects have different strong varying characteristics. Therefore, the adoption of the traceability strategy is a critical issue. Adopting a traceability strategy depends on time effort and organizational support.	5	[1], [6], [26], [60]
No/poor versioning support of traces: Trace links are evolving. Therefore, storing and managing different versions of trace links is still a challenge. Losing trace links means also losing requirements building information.	4	[3], [43]
No/poor reusability: The reusability of trace information is still a challenge but has the potential to save time. Only a few studies have been carried out in this area. They uncovered that only content oriented traces are the most promising basis for reusability.	2	[33], [51]

3.5 Solutions & Suggestions

Table 6 delivers the answers to RQ2 about the current state of problems and solutions in pre-RS traceability. The codings are based on general suggestions and applied techniques described in the literature. As for the previous table 5 the codes are ordered by descending number of codings, too.

Interestingly, the *person-related solutions* code is not the most frequently used code even though the person-related problems were identified as the most used code in the previous problem and challenges section 3.4. That aligns with suggestions in the literature to focus more on the human factor.

A lot of literature references *solutions regarding specific trace link specifications*. Figure 1 presents a simple model of a trace link including possible link and object types written in blue. The code *solutions regarding specific trace link specifications*, including child codes, consists of codings that developed a general model or slightly adapt it to meet project-specific needs. The literature presents varying task-related trace link types or solutions to support different origin object types like media data. Media data can be records, photos of sketches, or prototypes.

Fig. 1: Simple trace link model

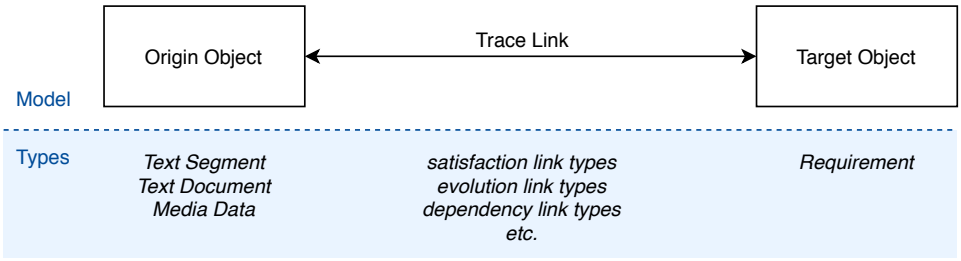

TABLE 6: Child codes of the *solutions & suggestions* code

Solutions & Suggestions: This code contains overviews of solutions. The child codes represent particular solutions.	1	[1]
Solutions regarding trace link specifications: This code summarizes solutions that define and use trace link specific information to trace between requirements and their origin.	GC	
Distinction of different trace link types: The usage of different trace link types is a common practice in RT in general and also in pre-RS traceability. But still, there is no common ground on a selection of trace link types. Distinguish between different trace link types supports filtering, querying, and analyzing for specific traces. Some mentioned trace link types can be used for pre- and post-RS traceability, like *satisfaction link types, evolution link types, dependency link types, etc*. Typical pre-RS traceability links are *rationals link types* and *contribution links*. The link type can also be an indicator of the persistence or volatility of trace links.	45	[1], [3], [7], [10], [18], [20], [21], [24], [26], [29], [30], [38], [45], [50], [51], [54], [60]
Support of non-functional requirements trace types: Frequently literature defines trace link types without focusing on functional or non-functional requirements. But still, some articles do handle non-functional trace links separately, for example, by special tags.	2	[3], [61]
Support different object types: The origin artifacts of requirements can be of different types like media objects (records from meetings or photos from sketches) or text documents. Supporting trace links to these different artifact types avoids transformations or transcriptions of media objects into one specific format. Tracing to such media types is also called *extended pre-traceability* and has the potential to support different project roles and increases the awareness of traceability needs.	6	[6], [12], [20], [33], [36]
Define trace link attributes: Trace links contain information like origin ID/name, target ID/name, and further attributes like link author, creation- and edit date, link version, artifact size, etc.	4	[3], [12], [51]
Person-related solutions: This code just groups all person-related solutions.	GC	
Support collaborative work & communication: Supporting collaboration and communication between project participants, including stakeholders, avoids the usage of other collaboration tools and therefore avoids the loss of requirements-specific information. Such a tool has to be easy-to-use for all users. Examples are web-based or community-centered applications. Also, tagging mechanisms to realize pre-RS traceability were mentioned in the literature to support collaboration.	8	[42], [51], [61], [62]
Support different involved (social) roles: Pre-RS traceability links artifacts created by different users with roles. Understanding these roles, their needs, and how they work with trace links and the artifacts is essential to find an appropriate pre-RS traceability strategy. The roles and their dependencies can be modeled and integrated into a tool. Taking care of these roles adds more information to trace links, and it simplifies the handling, accessing, and visualizing of the trace link information.	11	[4], [7], [9], [13], [19], [36], [62], [65]

Provide contact information: Providing contact information, including the job position or the project-related role is valuable. It potentially increases the awareness and importance of pre-RS traceability.	2	[9], [20]
Increase awareness of traceability need: Increasing the awareness of pre-RS traceability is essential and related to many other solution codes. Especially low-end users need more education in the needs and benefits of pre-RS traceability. Training or workshops should involve all project-related roles, including management roles. Furthermore, a well predefined traceability strategy and valuable information presentations support awareness. An additional solution is to establish a traceability responsibility role.	10	[1], [9], [10], [12], [24], [41], [43], [50]
Establish a new trace responsibility role: It is recommended to create a dedicated role for establishing, predefining, and managing traceability within a project or across different projects. This role has a managerial character. The role's focus is on process supervision and training of employees.	5	[1], [9], [10]
Provide immediate benefits to increase motivation: Immediate benefits can be realized through valuable feedback. Examples for valuable feedback are: doing peer reviews or providing access to organizational memory of traceability information to extract knowledge.	2	[3], [41]
Provide usage goal for traceability: Identifying goals and needs for traceability and trace links increases the awareness of pre-RS traceability. It is all about answering the question: "Why?".	2	[38]
Guide user to create trace link: Guiding the user creating a trace link can be achieved by rules about allowed link types for particular artifact types. Also, tagging mechanisms support users to create trace links because of their flexibility and lightweight. It is necessary to make it as easy as possible for the user.	7	[24], [34], [40], [50]
Support of flexible use: Flexible usage means letting the user access, filter, or create trace links and information in one or many ways to provide an optimal integration in the workflow of each individual user.	1	[1]
Need understanding by stakeholder: Especially high-end users realized that understanding by stakeholders supports the implementation and maintenance of pre-RS traceability. The understanding by stakeholders can be improved by training and workshops.	1	[41]
Obtain & record trace links/information: Obtaining and record trace information is a large amount of work. Therefore, creating trace links as a by-product is a popular approach. Further support can be provided by obtaining documents in the original format, obtaining trace links at the time of production (online), and providing tool support.	5	[1], [9], [36]
Recording types: The literature mentioned different types of how to obtain trace links. The child codes summarize these different approaches.	GC	
Record all available information: To avoid a lack of requirement origin information literature mentioned the recording of everything. Verbal communication should be recorded to keep track of it. Tool support is required to achieve this.	8	[1], [5], [33], [51], [53]
Priority traceability: The opposite approach to the code above *record all available information* suggests it is better to define what to trace and what not to trace. It prevents being overwhelmed by information. But this approach requires predefined rules or structures (models, ontologies, etc.).	2	[10], [28]
Support (semi-)automatic trace link capturing: The benefits of automation are obvious: the reduction of extra work-load. But still there exists no error-free (semi-)automatic trace capturing solution. For example, current solutions create candidate lists of trace links. These lists must be examined by analysts. The feedback of the analysts is used as input to improves the tracing algorithm. Ontologies or information retrieval methods can improve trace algorithms, too. Some approaches capture trace links in real-time and other approaches recover trace links.	9	[6], [12], [26], [37], [38], [51], [66]

Support large trace sets: The creation and maintenance of RS lead to a large amount of data that has to be traceable. Handling this growing amount of requirements, trace links, and origin artifacts is an essential task of a scalable traceability strategy. The relevance and priority of trace links and artifacts can be used to structure the data. The capability-based traceability approach or tagging mechanisms are examples of well scalable strategies.	5	[1], [24], [34], [37], [50]
Predefined trace information: Predefine what and how to trace answers provides clarifications, for example, about trace link granularity. The predefinitions can also be reused and shared with other projects.	5	[10], [19], [24], [38], [51]
Description-fields on requirement: The so-called "rationale" is added to a requirement to clarify the reason behind it. The rationale-attribute is requirement-specific and has to be maintained.	2	[42]
Create/improve the structure of RS: Structuring the RS supports the understanding, searching, and managing of requirements. It supports pre-RS traceability by restricting traceability to a subset of high-level or business-critical requirements.	1	[42]
Hierarchical structuring of requirements: The *refinement hierarchy* unites all requirements for one functionality on different levels of detail. The *aspect hierarchy* unites requirements involved in one particular relationship and therefore supports different viewpoints. So one requirement can belong to more than one aspect.	6	[33], [42], [46], [67]
Classification/categorization of requirements: Classifications or categorizations of requirements can be used by single assignments or by adding them into multiple orthogonal categories.	3	[68], [69]
Highlight of key requirements: The concept of key requirements used on the stakeholder level is also called KURs (“Key User Requirements”) or KPIs (“Key Performance Indicators”). Key requirements are a small set of abstract requirements that describe the essence of the system. They contain the reason why the stakeholder wants to buy the system/application.	3	[42]
Grading of requirements according to importance: Requirements can be graded in different ways. One approach grades and prioritizes requirements concerning the stakeholder requirements. Another approach describes the usage of a platform for discussing and rating requirements. The distinction of requirements based on the level of importance supports selective traceability. For example, high-end users only trace mission-critical requirements to keep costs under control.	3	[41], [62], [69]
Provide flexible tools/support: Tool support is necessary to handle a large amount of data and to support traceability activities. A tool has to be customizable for different project environments. Often the integration of more project-specific tools is necessary. To ensure the usage of the tool by the user, a simple and understandable user interface must be available. Further characteristics of a tool-support are important like flexibility, accessibility for all involved, etc.	16	[1], [3], [37], [39]–[41], [43], [48], [51]
Enable valuable visualizations & presentations: The advantages of data visualizations and presentations are well known and also valuable for pre-RS traceability. Visualizing and presenting trace links support analyzing, browsing, and filtering of information. Furthermore, traceability information is better accessible and understandable.	16	[1], [3], [7], [9], [12], [33], [37], [40], [46]
Improve generic trace model: Providing and sharing a traceability model avoids extra effort and guidance to the user. To support the reusability of the generic model it is conductive to store it separately.	4	[6], [38]
Define a traceability schema: Establishing a traceability schema as early as possible improves long-term traceability. Such a schema should contain defined trace link types, linkage rules, permissions about accessibility and security, guidelines for maintenance, levels of trace granularity, etc.	8	[3], [5], [7], [38], [56]
Define trace preconditions: To successfully establish traceability following preconditions have to be defined: one or more responsible persons, traceability rules, and training to increase the awareness.	3	[10], [38], [49]

Provide fitting storage for trace links and artifacts: An often-used approach describes the separate storing of trace links and artifacts. This separate traceability storage needs unique identifiers to manage links and artifacts.	4	[26], [42]
Establish central storage/repository: Many tools use an underpinned central storage and a generic interface. This central storage provides persistence as a single point of truth. A traceability repository is one example of central storage that also supports the versioning of traces.	9	[1], [5], [26], [43], [51], [67]
Organize individual (origin) artifacts: Various approaches exist to organize different origin artifacts like documents, media objects, etc. One solution uses a wiki-based tool to store the artifacts. Another solution creates formalized or semi-formalized representations. They are better to process but are not or poorly understandable for stakeholders. The literature also mentions a combination of different integrated tools or the provision of a generic interface for accessing the artifacts. Organizing all these origin artifacts is related to code *obtaining & recording of trace information* and enables of valuable visualizations & presentations.	10	[1], [12], [33], [55]
Maintain trace links & artifacts: The maintenance of artifacts and their trace links is essential for further usage. The literature recommends to provide tool support and to define maintenance activities by methods or guidelines. Furthermore, establishing new responsibility roles is also recommended.	2	[1], [9]
Design a modular viable (traceability)system: Supporting an evolving system can be achieved by modular architecture. The same applies to artifacts and their trace links.	3	[1], [24]
Establish periodically audits/quality checks: Establishing regular audits keeps the completeness and consistency of artifacts and their trace links on a high enough level.	1	[10]
Notify responsible persons in case of changes: Notify one responsible person or all involved persons in case of changes is a recommended practice to keep artifacts and trace links up-to-date.	1	[13]
Establish templates & guidelines: The literature recommends establishing standards, guidelines, and best practices. They are used for different purposes like formulating requirements, defining traceable artifacts, or the management of individual artifacts.	7	[1], [3], [9], [34], [41], [42]
Configurable traceability strategies: Each project is unique in collaboration, artifacts, participants, project type, etc. Therefore, it is necessary to customize the pre-RS traceability strategy.	3	[1], [51]
Analyse & consider special attributes: The RS and most of the origin documents are written in natural language. Therefore, natural language processing can be used to uncover trace links and relationships. Language-specific characteristics like synonyms are typical pre-RS traceability problems. This must be taken into account when finding a suitable natural language processing (NLP) technique.	2	[2]
Support information transfer: It is valuable to make the work process and its responsibilities visible and to write explicit and shared RS to achieve a smooth information transfer (internal and external). Pre-RS traceability supports visibility and transparency and potentially can support information transfer.	1	[1]

3.6 Tools & Techniques

Table 7 presents the codes related to RQ3 about existing tools and techniques to support pre-RS traceability. The main codes *traceability tools* and *traceability techniques* are used to reference literature presenting overviews or comparisons.

The code *traceability tools* summarizes tools that support pre-RS traceability in any way. We do not provide a short description of each tool here, as this can be found in the referenced literature. The referenced tools are ordered by descending number of codings. This code order identifies DOORS from IBM as the most referenced tool. DOORS is known for managing requirements, including RT, and is well established in the industry. Another tool like Pro-ART focuses on pre-RS traceability. This tool is well known in science but to the best of our knowledge not used in the industry. We observe that tools established in the industry are either (a) very adaptable and support many tasks of the development process (like DOORS) or (b) are implemented specifically for the project environment, including the integration into the existing tooling landscape (like MARS). For the child codes of *traceability techniques*, we used the order according to a

logical sequence. The first child code *trace model aspects* refers to the general required activities that have to be considered in each tool and technique: definition, production, and extraction of trace links.

The *distinction based on traceability types* code consists of frequently mentioned traceability types ordered by descending codings. We identified *personnel-based RT* as a type referenced more often. Many pre-RS traceability techniques based on more than one traceability type. They often use one predominant type and at least one subordinate type. The next five codes *connection by codes/words/tags*, *model driven development (MDD)*, *natural language processing (NLP)*, *ontology* and *graphs* are reference to further techniques which can be combined with each other or with child codes of the *traceability types* code.

TABLE 7: Sub codes of the *tools* and Traceability models/techniques codes

Traceability tools:	3	[8], [11]
Tools from IBM:	GC	
DOORS [3], [11], [18], [37], [42], [43]; Rational RequisitePro [11], [37]; MARS based on Focal Point the Main Requirement Specification (MRS) [11]		
Requirements tracing on-target (RETRO) [11], [22], [37]; Traceability of 0bject-0riented Requirements (TOOR) [18], [53]; Business Insight Toolkit (BITKit) [39], [40]; Advanced Multimedia Organizer for Requirements Elicitation (AMORE) [33]; Echo [5]; Pro-ART [18], [51]; Information Engineering Facility (IEF) [1]; Requirements Traceability Manager (RTM) [1]; Integrated System Design TOOL (ISDT) [7]; SuperTracePlus (STP) [37]; TRAM [11]; DesignTrack [11]; Teamcenter Systems Engineering tool (TcSE) [46]; Tool combinations [61]		
Traceability techniques: This code contains overviews of traceability types. The child codes represent particular traceability techniques.	3	[8], [25]
Trace model aspects: Each traceability model or technique has to consider the three aspects covered by the child codes: trace definition, trace production, and trace extraction.	1	[6]
Definition of trace links: A trace link definition contains all elements to include in a link like an origin artifact, target artifact, and trace link. It also includes the different types of these elements and how they will be stored and used.	3	[6]
Production of trace links: The link production contains information about how and when capturing a trace link, how to precept and register it, and how to maintain it.	5	[3], [6], [41]
On-line: Capturing trace links on-line means storing links automatically during creation or changing artifacts as a by-product. In the case of pre-RS traceability, the trace link to the origin document will be stored during creating or maintaining a requirement.	4	[3], [43]
Off-line: Recording a trace link off-line means storing the link automatically or manually after the origin and target artifact was produced.	1	[3]
Extraction of trace links: A traceability model should support flexible and varying use cases. Extraction mechanisms are selective tracing (filtering traces by selected patterns or characteristics), interactive tracing (browsing, guiding and navigating through a set of trace links), and non-guided tracing (going from one artifact or a trace link to another at will).	2	[6]
Distinction based on traceability types: This code contains different traceability types also used for pre-RS traceability. The distinction between these types does not mean that specific techniques do not combine different types. But at least one traceability type is mostly predominant.	GC	
Personnel-based RT (PBRT): According to the frequently used *person-related problem* code, varying PBRT techniques exist. These techniques focus on person-related behavior and individual intentions.	5	[4], [19]

Contribution structure: The contribution structure is a common and frequently referenced technique that focuses on pre-RS traceability. The contribution structure combines personnel-based and artifact-based RT. This technique distinguishes between three agent types (principal agent, author agent, and the documentor agent) to capture them for each artifact. The artifact-base RT is realized by three relations: temporal relations (provides chronological order and requirements history), developmental relations (provides logical order), and auxiliary relations (provides additional types of order).	14	[4], [6], [21], [44], [47], [52]
Agent-based knowledge: This solution supports both, pre- and post-RS traceability. The technique supports organizational changes by analyzing and tracking agent's interventions. It combines personnel-based and goal-based RT.	9	[27]
Usage-centered technique approach: Constructing a system based on the usage pattern is called a usage-centered design technique. One approach uses this technique to build a system that can trace user interface design decisions and tasks.	3	[36]
Simple traceability links: Simple linking techniques store the trace links between artifacts explicitly. The child codes present two examples.	2	[31], [42]
RT matrix: The RT matrix is a common approach to trace between artifacts. It is a simple option to visualize many-to-many relationships. But the RT matrix becomes confusing and hard to maintain in large projects.	5	[10], [37], [42], [47]
Hyper-linked documents: Highlighted statements in documents can be traced by hyper-links. Traversing these links physically to see all information can be time-consuming. Additionally, it is hard to recognize broken links with already deleted start or target items.	1	[42]
Goal-centric traceability (GCT): It is also called goal-oriented requirements engineering (GORE). In the case of pre-RS traceability, it supports trace links between requirements, stakeholders, and project goals. GCT can realize the tracing of functional and non-functional requirements. Much further research is possible in analyzing social issues and the interaction of social roles to uncover goals.	8	[11], [27], [44], [54]
Feature-oriented RT (FORT): FORT is used for tracing functional requirements. It is a type of selective traceability based on feature prioritization. FORT caused a reduction of trace link generation effort with 24-72% [11].	5	[11], [68]
Value-based RT (VBRT): Like FORT, VBRT is used for functional traceability. This technique distinguish between requirements that are valuable to trace and requirements with less value to trace. This type of selective tracing reduces the effort with 35% compared to full tracing.	5	[11]
Event-based RT (EBT): EBT is based on the "publish-subscribe" mechanism and handles functional and non-functional requirements. This technique supports maintenance by creating links after a change request executed.	5	[11], [31]
Rule-based approaches (RB): Reducing cost and increasing efficiency can be achieved by using RB approaches. RB support functional and non-functional requirements. It is based on predefined rules on structures, classifications, and natural language processing.	3	[3], [11]
Artifact-based RT: This RT technique is based on existing relationships between artifacts or different versions of artifacts.	3	[4]
Knowledge-based techniques: Knowledge-based traceability uncovered relationships and potential change impacts based on historical changes.	1	[31]
Connection by codes/words/tags: This technique consists of origin and target artifacts or statements linked by one word, code, or tag. This technique has become particularly popular in recent times, as it is a lightweight and well scalable approach.	GC	

Qualitative data analysis (QDA): QDA is a common scientific theory building method to analyze unstructured qualitative data. QDA results in a code system that structures uncovered concepts. Techniques using QDA link statements inside origin artifacts with domain model elements or requirements by codes of the code system.	19	[34], [70]
Tagging: This technique allows the tagging of artifacts. The relationships between two or more artifacts are created by using the same tag or defined relationships between particular tags.	10	[37], [40], [50], [65]
Capabilities-based development: This technique links statements inside artifacts of the problem space and requirements of the solution space by capabilities.	7	[11], [24]
Model driven development (MDD): Using MDD is a popular solution to support traceability, including pre- and post-RS traceability. The literature presents different varying modeling solutions, like modeling goals, domains, or roles. The MDD can be combined with other techniques like information retrieval (IR). The child codes present different MDD solutions.	1	[23]
Meta models: Traceability meta-models often supporting both pre- and post-RS traceability. Meta-models are a way to provide automatization for trace link capturing. Particular types of meta-models are design-centered models (modeling a framework) or database guided models (modeling how to register a trace in a database).	24	[3], [6], [18], [19], [26], [29], [30], [38], [51], [60]
Concept models: A concept model links different concepts or entities like roles, requirements, or product lines.	8	[3], [67]
Hypertext models: These models are created to capture informal or unstructured information, particularly in source documents.	1	[6]
Richer traceability model: This technique traces RE artifacts back to social interactions based on a graph like representation.	1	[44]
Natural language processing (NLP): NLP is suitable for processing unstructured information like interviews, protocols, or requests. Therefore, it is useful to support pre-RS traceability. The child codes present explicit information retrieval (IR) and NLP techniques.	16	[3], [11], [12], [22], [23], [28], [31], [37], [55], [58], [66]
Latent semantic analysis (LSA): The specialty of LSA is to determine semantic equivalence between synonyms and whole documents.	3	[2]
Shallow natural language processing: This technique is based on statistical properties of language structure rather than models or absolute logical rules.	1	[22]
Ontology: Ontologies are a common technique to represent knowledge by modeling requirements, goals, tags, etc. Traceability taxonomies can be derived from ontologies. They are suitable for large and complex systems.	9	[11], [23], [28], [61]
Graphs: Graphs are used for pre-RS traceability in different variants and by different techniques and tools, for example, to link different levels of abstraction, interactions, or dependencies.	5	[19], [23], [28], [44]

4 Discussion

This technical report presents the resulting code system of the qualitative data analysis (QDA) on pre-requirements specification (pre-RS) traceability based on three research questions.

The four most used main codes that emerged in the code system mirror the objectives of the three research questions, with problems and solutions (Q2) being split into two categories. The least used category, however, was surprising. We considered ”consequences of poor pre-RS traceability important enough to warrant it being one of the nine main categories. Yet, in the literature, we reviewed we only found evidence in four instances of codings. This suggests that the consequences of neglecting this part of requirements traceability (RT) may need more explicit study.

Previous systematic literature reviews (SLR) focus on RT and merely mentioned pre-RS traceability as a sub-topic. Other literature consists of specially developed solutions to solve the pre-RS traceability problem. But still, there is no general solution. To get an overview of the state-of-the-art in pre-RS traceability we conducted a SLR. We present the whole code system of the QDA in this technical report. The main goal of this report is to provide related literature to pre-RS traceability related topics for researchers who want to know more about particular codes of the code system. Additionally,

we are working on a separate SLR paper to provide a comprehensive overview of the state-of-the-art of pre-RS traceability including all findings of our analysis and detailed answers to our research questions.

Pre- and post-RS traceability as parts of RT are strongly related to each other. Therefore, it was difficult to extract only pre-RS traceability related information. Finally, the information included in the code system is either explicitly about pre-RS traceability or applies to this topic in particular as well as to RT in general.

During our analysis, we uncovered 26 techniques and ideas supporting pre-RS traceability (section 3.6) which are evaluated by single case-, field study, or a demonstration. Three techniques are evaluated by multiple case case studies or multiple data sets [37], [53], [55]. Only two papers reported on elaborate evaluations [24], [52]. Excluding non-evaluated literature would, therefore, lead to a significant loss of information. To provide a comprehensive overview of pre-RS traceability we include this information. Our following SLR will also refer to this problem in detail. We are working on an overview to present these different evaluation states.

Even though we have found plenty of benefits and use cases laid out repeatedly in the literature (section 3.3) still there exists a large gap between scientific solutions and industry practice for pre-RS traceability [3], [37]. With respect to all research in the area of pre-RS traceability, not enough evaluation of developed solutions is seen as one reason. Furthermore, most case studies do not correspond to the size of real projects [3], [37]. Much more research has to be done in analyzing current industry practices and how new solutions can be applied. However, it is not only science that should be accommodating to the industry, but the industry must also be open to new approaches and the corresponding additional effort to apply them.

5 Conclusion

Pre-requirements specification (pre-RS) traceability links requirements with their origin. The knowledge about requirements creation and who is involved in it has a significant impact on project success. Compared to post-RS traceability significantly less research was done on pre-RS traceability. Pre-RS traceability is frequently mentioned as a sub-topics in previous systematic literature reviews (SLR) about requirements traceability (RT). Therefore, we will conduct a SLR about pre-RS traceability to provide an overview of the state-of-the-art. This technical report is the first step in presenting our results.

To structure our research we developed three research questions to capture information about the main categories of relevance, problems, and current solutions. Analyzing the relevance of pre-RS traceability is important to motivate this topic because it is often seen just as a RT activity with high effort and comparatively small benefits.

We used qualitative data analysis (QDA) to process 67 relevant papers. This technical report presents the resulting code system of our analysis. The code system contains information and related references to the literature about use cases, benefits, problems, solutions, and current pre-RS traceability tools and techniques. This technical report serves researches who want to dive into more details of the pre-RS traceability topic. We provide a comprehensive overview of the related work in the field of pre-RS traceability, mapped to the focus of our research questions. We point out potential areas of future research where we found gaps in the analysis of recent scientific literature. To present a comprehensive overview of the topic we are working on the SLR paper. This SLR paper will present all findings and explicit answers to our research questions.

References

[1] O. C. Z. Gotel, "Requirements Traceability," Oxford, Main Report, 1992.

[2] A. Stone and P. Sawyer, "Identifying tacit knowledge-based requirements," *IEE Proceedings - Software*, vol. 153, no. 6, pp. 211–218, Dec. 2006.

[3] S. Winkler and J. von Pilgrim, "A survey of traceability in requirements engineering and model-driven development," *Software & Systems Modeling*, vol. 9, no. 4, pp. 529–565, Sep. 2010.

[4] O. Gotel and A. Finkelstein, "Revisiting requirements production," *Software Engineering Journal*, vol. 11, no. 3, pp. 166–182, May 1996.

[5] C. Lee, L. Guadagno, and X. Jia, "An agile approach to capturing requirements and traceability," in *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2003)*, vol. 20, 2003.

[6] F. A. C. Pinheiro, "Requirements Traceability," in *Perspectives on Software Requirements*, ser. The Springer International Series in Engineering and Computer Science, J. C. S. do Prado Leite and J. H. Doorn, Eds. Boston, MA: Springer US, 2004, pp. 91–113.

[7] V. Shukla, G. Auriol, and C. Baron, "Integrated requirement traceability, multiview modeling, and decision-making: A systems engineering approach for integrating processes and product," in *2012 IEEE International Systems Conference SysCon 2012*. Vancouver, BC, Canada: IEEE, Mar. 2012, pp. 1–5.

[8] H. Tufail, M. F. Masood, B. Zeb, F. Azam, and M. W. Anwar, "A systematic review of requirement traceability techniques and tools," in *2017 2nd International Conference on System Reliability and Safety (ICSRS)*. Milan, Italy: IEEE, 2017.

[9] O. C. Z. Gotel and C. W. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of IEEE International Conference on Requirements Engineering*, Apr. 1994, pp. 94–101.

[10] K. E. Wiegers and J. Beatty, *Software Requirements*. Redmond: Microsoft Press, 2013.

[11] R. Torkar, T. Gorschek, R. Feldt, M. Svahnberg, R. Uzair Akbar, and K. Kamran, "Requirements Traceability : A Systematic Review and Industry Case Study," *International Journal of Software Engineering and Knowledge Engineering*, vol. 22, no. 3, pp. 385–433, 2012.

[12] S. Altaf, A. Shah, N. Imtiaz, A. S. Shah, and S. F. Ahmed, "Visualization representing benefits of pre-requirement specification traceability," *International Journal of Engineering & Technology*, vol. 7, p. 44, 2018.

[13] R. Wohlrab, J.-P. Steghöfer, E. Knauss, S. Maro, and A. Anjorin, "Collaborative Traceability Management: Challenges and Opportunities," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, Sep. 2016, pp. 216–225.

[14] P. Grunbacher, M. Halling, S. Biffl, H. Kitapci, and B. Boehm, "Repeatable quality assurance techniques for requirements negotiations," in *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*. Big Island, HI, USA, USA: IEEE, Jan. 2003, pp. 9 pp.–.

[15] B. Kitchenham, “Procedures for Performing Systematic Reviews,” Keele, Technical Report 1.0, Jul. 2004.
[16] A. Strauss and J. Corbin, *Basics of qualitative research*. Sage publications, 1990.
[17] S. Spall, “Peer Debriefing in Qualitative Research: Emerging Operational Models,” *Qualitative Inquiry*, vol. 4, no. 2, pp. 280–292, Jun. 1998.
[18] B. Ramesh, C. Stubbs, T. Powers, and M. Edwards, “Requirements traceability: Theory and practice,” *Annals of Software Engineering*, vol. 3, no. 1-2-3-4, pp. 397–415, 1997.
[19] J. Castro, R. Pinto, A. Castor, and J. Mylopoulos, “Requirements Traceability in Agent Oriented Development,” in *Software Engineering for Large-Scale Multi-Agent Systems*, ser. Lecture Notes in Computer Science, A. Garcia, C. Lucena, F. Zambonelli, A. Omicini, and J. Castro, Eds. Berlin, Heidelberg: Springer, 2002, pp. 57–72.
[20] A. Ahmad and M. A. Ghazali, “Documenting Requirements Traceability Information for Small Projects,” in *2007 IEEE International Multitopic Conference*. Lahore, Pakistan: IEEE, Dec. 2007, pp. 1–5.
[21] O. Gotel and A. Finkelstein, “Contribution structures [Requirements artifacts],” in *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE’95)*. York, UK, UK: IEEE, Mar. 1995, pp. 100–107.
[22] P. Sawyer, R. Gacitua, and A. Stone, “Profiling and Tracing Stakeholder Needs,” in *Innovations for Requirement Analysis. From Stakeholders’ Needs to Formal Designs*, ser. Lecture Notes in Computer Science, B. Paech and C. Martell, Eds. Monterey, CA, USA: Springer, 2007, pp. 196–213.
[23] K. Souali, O. Rahmaoui, and M. Ouzzif, “An overview of traceability: Definitions and techniques,” in *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*, Oct. 2016, pp. 789–793.
[24] R. Ravichandar, J. D. Arthur, and M. Pérez-Quiñones, “Pre-Requirement Specification Traceability: Bridging the Complexity Gap through Capabilities,” *International Symposium on Grand Challenges in Traceability, TEFSE/GCT 2007*, p. 10, Mar. 2007.
[25] M. F. Bashir and M. A. Qadir, “Traceability Techniques: A Critical Study,” in *2006 IEEE International Multitopic Conference*, Dec. 2006, pp. 265–268.
[26] G. Spanoudakis and A. Zisman, “Software traceability: a roadmap,” in *Handbook of Software Engineering and Knowledge Engineering*. WORLD SCIENTIFIC, Aug. 2005, pp. 395–428.
[27] G. Urrego-Giraldo, “Agent-based knowledge keep tracking,” in *Proceedings Fifth IEEE Workshop on Mobile Computing Systems and Applications*. Las Vegas, NV, USA, USA: IEEE, Oct. 2003, pp. 8–16.
[28] K. Souali, O. Rahmaoui, and M. Ouzzif, “An Overview of Traceability: Towards a general multi-domain model,” *Advances in Science, Technology and Engineering Systems Journal (ASTES)*, vol. 2, no. 3, pp. 356–361, 2017.
[29] Y. He and X. Li, “RE_prov: Modeling Requirement Provenance with PROV,” in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. Hamilton, New Zealand: IEEE, Dec. 2016, pp. 397–400.
[30] H. Dubois, M.-A. Peraldi-Frati, and F. Lakhal, “A Model for Requirements Traceability in a Heterogeneous Model-Based Design Process: Application to Automotive Embedded Systems,” in *2010 15th IEEE International Conference on Engineering of Complex Computer Systems*. Oxford, UK: IEEE, Mar. 2010, pp. 233–242.
[31] S. Imtiaz, N. Ikram, and S. Imtiaz, “Impact Analysis from Multiple Perspectives: Evaluation of Traceability Techniques,” in *2008 The Third International Conference on Software Engineering Advances*. Sliema, Malta: IEEE, Oct. 2008, pp. 457–464.
[32] A. Ferrari, F. dell’Orletta, G. O. Spagnolo, and S. Gnesi, “Measuring and Improving the Completeness of Natural Language Requirements,” in *Requirements Engineering: Foundation for Software Quality*, ser. Lecture Notes in Computer Science, C. Salinesi and I. van de Weerd, Eds. Springer International Publishing, 2014, pp. 23–38.
[33] D. Wood, M. Christel, and S. Stevens, “A multimedia approach to requirements capture and modeling,” in *Proceedings of IEEE International Conference on Requirements Engineering*. Colorado Springs, CO, USA, USA: IEEE, Apr. 1994, pp. 53–56.
[34] A. Kaufmann and D. Riehle, “Improving Traceability of Requirements Through Qualitative Data Analysis,” in *Proceedings of the Software Engineering 2015*, 2015.
[35] K. Mohan and B. Ramesh, “Managing variability with traceability in product and service families,” in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*. Big Island, HI, USA: IEEE, Jan. 2002, pp. 1309–1317.
[36] H. M. Hao and A. Jaafar, “Tracing user interface design pre-requirement to generate interface design specification,” in *2009 International Conference on Electrical Engineering and Informatics*, vol. 01. Selangor, Malaysia: IEEE, Aug. 2009, pp. 287–292.
[37] J. Hayes, A. Dekhtyar, and S. Sundaram, “Advancing candidate link generation for requirements tracing: the study of methods,” *IEEE Transactions on Software Engineering*, vol. 32, no. 1, pp. 4–19, Jan. 2006.
[38] S. Haidrar, A. Anwar, and O. Roudies, “Towards a generic framework for requirements traceability management for SysML language,” in *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*. Tangier, Morocco: IEEE, Oct. 2016, pp. 210–215.
[39] H. Ossher, R. Bellamy, I. Simmonds, D. Amid, A. Anaby-Tavor, M. Callery, M. Desmond, J. de Vries, A. Fisher, and S. Krasikov, “Flexible modeling tools for pre-requirements analysis: conceptual architecture and research challenges,” in *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*, ser. OOPSLA ’10. Reno/Tahoe, Nevada, USA: Association for Computing Machinery, Oct. 2010, pp. 848–864.
[40] H. Ossher, R. Bellamy, D. Amid, A. Anaby-Tavor, M. Callery, M. Desmond, J. de Vries, A. Fisher, T. Frauenhofer, S. Krasikov, I. Simmonds, and C. Swart, “Business insight toolkit: Flexible pre-requirements modeling,” in *2009 31st International Conference on Software Engineering - Companion Volume*. Vancouver, BC, Canada: IEEE, May 2009, pp. 423–424.
[41] B. Ramesh, “Factors Influencing Requirements Traceability Practice,” *Communications of the ACM*, vol. 41, no. 12, pp. 37–44, 1998.
[42] J. Dick, E. Hull, and K. Jackson, *Requirements Engineering*, 4th ed. Switzerland: Springer International Publishing, 2017.
[43] P. Mader, O. Gotel, and I. Philippow, “Motivation Matters in the Traceability Trenches,” in *2009 17th IEEE International Requirements Engineering Conference*, Aug. 2009, pp. 143–148.
[44] M. Serrano and J. C. S. do Prado Leite, “A Rich Traceability Model for Social Interactions,” in *Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*, ser. TEFSE ’11. New York, NY, USA: ACM, 2011, pp. 63–66.
[45] B. Ramesh and M. Jarke, “Toward reference models for requirements traceability,” *IEEE Transactions on Software Engineering*, vol. 27, no. 1, pp. 58–93, Jan. 2001.
[46] M. C. Panis, “Successful Deployment of Requirements Traceability in a Commercial Engineering Organization...Really,” in *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, ser. RE ’10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 303–307.
[47] S. Jayatilleke and R. Lai, “A systematic review of requirements change management,” *Information and Software Technology*, vol. 93, pp. 163–185, Jan. 2018.
[48] E. Bouillon, P. Mäder, and I. Philippow, “A Survey on Usage Scenarios for Requirements Traceability in Practice,” in *Requirements Engineering: Foundation for Software Quality*, ser. Lecture Notes in Computer Science, J. Doerr and A. L. Opdahl, Eds. Berlin, Heidelberg: Springer, 2013, pp. 158–173.
[49] A. Finkelstein, “Requirements engineering: a review and research agenda,” in *Proceedings of 1st Asia-Pacific Software Engineering Conference*. Tokyo, Japan, Japan: IEEE, Dec. 1994, pp. 10–19.
[50] H. Ossher, D. Amid, A. Anaby-Tavor, R. Bellamy, M. Callery, M. Desmond, J. De Vries, A. Fisher, S. Krasikov, I. Simmonds, and C. Swart, “Using tagging to identify and organize concerns during pre-requirements analysis,” in *2009 ICSE Workshop on Aspect-Oriented Requirements Engineering and Architecture Design*. Vancouver, BC, Canada: IEEE, May 2009, pp. 25–30.
[51] K. Pohl, “PRO-ART: enabling requirements pre-traceability,” in *Proceedings of the Second International Conference on Requirements Engineering*, Apr. 1996, pp. 76–84.

[52] O. Gotel and A. Finkelstein, "Extended requirements traceability: results of an industrial case study," in *Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*. Annapolis, MD, USA, USA: IEEE, Jan. 1997, pp. 169–178.
[53] F. Pinheiro and J. Goguen, "An object-oriented tool for tracing requirements," *IEEE Software*, vol. 13, no. 2, pp. 52–64, Mar. 1996.
[54] P. Rempel, P. Mäder, and T. Kuschke, "An empirical study on project-specific traceability strategies," in *2013 21st IEEE International Requirements Engineering Conference (RE)*. Rio de Janeiro, Brazil: IEEE, Jul. 2013, pp. 195–204.
[55] H. Kitapci and B. W. Boehm, "Formalizing Informal Stakeholder Decisions–A Hybrid Method Approach," in *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*. Waikoloa, HI, USA: IEEE, Jan. 2007, pp. 283c–283c.
[56] A. Espinoza, P. P. Alarcon, and J. Garbajosa, "Analyzing and Systematizing Current Traceability Schemas," in *2006 30th Annual IEEE/NASA Software Engineering Workshop*, Apr. 2006, pp. 21–32.
[57] G. R., M. L., N. B., P. P., d. R. A. N., R. M., S. P., W. A., and Y. H., "Making Tacit Requirements Explicit," in *2009 Second International Workshop on Managing Requirements Knowledge*. Atlanta, GA, USA: IEEE, Sep. 2009, pp. 40–44.
[58] A. Stone and P. Sawyer, "Exposing Tacit Knowledge via Pre-Requirements Tracing," in *14th IEEE International Requirements Engineering Conference (RE'06)*. Minneapolis/St. Paul, MN, USA: IEEE, Sep. 2006, pp. 353–354.
[59] J. Leite and A. Oliveira, "A client oriented requirements baseline," in *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)*. York, UK, UK: IEEE, Mar. 1995, pp. 108–115.
[60] H. El Ghazi and S. Assar, "A multi view based traceability management method," in *2008 Second International Conference on Research Challenges in Information Science*. Marrakech, Morocco: IEEE, Jun. 2008, pp. 393–400.
[61] P. Liang, P. Avgeriou, K. He, and L. Xu, "From collective knowledge to intelligence: pre-requirements analysis of large and complex systems," in *In: Proceedings of the 1st Workshop on Web 2.0 for Software Engineering (Web2SE), ACM*, 2010, pp. 26–30.
[62] S. Lohmann, S. Dietzold, P. Heim, and N. Heino, "A web platform for social requirements engineering," in *Software Engineering 2009 - Workshopband*. Gesellschaft für Informatik e.V., 2009, accepted: 2019-02-20T10:13:00Z ISSN: 1617-5468.
[63] O. C. Z. Gotel and S. J. Morris, "Out of the labyrinth: Leveraging other disciplines for requirements traceability," in *2011 IEEE 19th International Requirements Engineering Conference*. Trento, Italy: IEEE, Aug. 2011, pp. 121–130.
[64] H. Kitapci and B. W. Boehm, "Using a Hybrid Method for Formalizing Informal Stakeholder Requirements Inputs," in *Fourth International Workshop on Comparative Evaluation in Requirements Engineering (CERE'06 - RE'06 Workshop)*. Minneapolis, MN, USA: IEEE, Sep. 2006, pp. 48–59.
[65] J. H. Weber-Jahnke and A. Onabajo, "Finding Defects in Natural Language Confidentiality Requirements," in *2009 17th IEEE International Requirements Engineering Conference*. Atlanta, GA, USA: IEEE, Aug. 2009, pp. 213–222.
[66] J. Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval," in *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.* Monterey Bay, CA, USA, USA: IEEE, Sep. 2003, pp. 138–147.
[67] J. Botaschanjan, A. Fleischmann, and M. Pister, "A Conceptual Model for Requirements Engineering and Management for Change-intensive Software." ACTA Press, Feb. 2004.
[68] P. Laurent, J. Cleland-Huang, and C. Duan, "Towards Automated Requirements Triage," in *15th IEEE International Requirements Engineering Conference (RE 2007)*. Delhi, India: IEEE, Oct. 2007, pp. 131–140.
[69] J. R. Jiao and C.-H. Chen, "Customer Requirement Management in Product Development: A Review of Research Issues," *Concurrent Engineering*, vol. 14, no. 3, pp. 173–185, Sep. 2006.
[70] A. Kaufmann and D. Riehle, "The QDAcity-RE method for structural domain modeling using qualitative data analysis," *Requirements Engineering*, vol. 24, no. 1, pp. 85–102, Mar. 2019.