Published in Proc. of the 21st IEEE International Requirements Engineering Conference (RE), Rio de Janeiro, Brazil
@INPROCEEDINGS{conf/re/RempelMK13, author = {Patrick Rempel and M{\"a}der, Patrick and Tobias Kuschke},
title = {An Empirical Study on Project-Specific Traceability Strategies},
booktitle = {Proc. of the 21st IEEE International Requirements Engineering Conference {(RE)}, Rio de Janeiro, Brazil}, year = {2013}}

An Empirical Study on Project-Specific Traceability Strategies

Patrick Rempel, Patrick Mäder, and Tobias Kuschke
Ilmenau Technical University
Department of Software Systems
Ilmenau, Germany
{patrick.rempel|patrick.maeder|tobias.kuschke}@tu-ilmenau.de

***Abstract*—Effective requirements traceability supports practitioners in reaching higher project maturity and better product quality. Researchers argue that effective traceability barely happens by chance or through ad-hoc efforts and that traceability should be explicitly defined upfront. However, in a previous study we found that practitioners rarely follow explicit traceability strategies. We were interested in the reason for this discrepancy. Are practitioners able to reach effective traceability without an explicit definition? More specifically, how suitable is requirements traceability that is not strategically planned in supporting a project's development process. Our interview study involved practitioners from 17 companies. These practitioners were familiar with the development process, the existing traceability and the goals of the project they reported about. For each project, we first modeled a traceability strategy based on the gathered information. Second, we examined and modeled the applied software engineering processes of each project. Thereby, we focused on executed tasks, involved actors, and pursued goals. Finally, we analyzed the quality and suitability of a project's traceability strategy. We report common problems across the analyzed traceability strategies and their possible causes. The overall quality and mismatch of analyzed traceability suggests that an upfront-defined traceability strategy is indeed required. Furthermore, we show that the decision for or against traceability relations between artifacts requires a detailed understanding of the project's engineering process and goals; emphasizing the need for a goal-oriented procedure to assess existing and define new traceability strategies.**

***Index Terms*—requirements traceability; traceability strategy; traceability strategy assessment, interview study; empirical study; traceability usage goals; project-specific traceability;**

I. Introduction

Effective requirements traceability supports practitioners in reaching higher project maturity and better product quality. Requirements traceability, defined as the ability to follow the life of a requirement in both a backward and forward direction [1] is a critical element of any rigorous software development process. It provides support for numerous software engineering tasks such as requirements validation, impact analysis, coverage analysis, compliance verification, and derivation analysis.

Researchers argue that effective traceability barely happens by chance or through ad-hoc efforts and that traceability should be explicitly defined upfront [2]. Such an explicit traceability strategy should at least define the artifacts to be traced and the traces to be created among them. These simple requirements conceal complex decisions as to the granularity, categorization, and storage of assorted multi-media artifacts [2]. However, in a previous study we found that practitioners rarely follow explicit traceability strategies [3]. Accordingly, we were interested in the reason for this discrepancy. Are practitioners able to reach effective traceability without an explicit definition? More specifically, how suitable is requirements traceability that is not strategically planned in supporting a projects development process.

We followed a qualitative research approach to answer our questions. Our interview study involved practitioners from 17 companies. These practitioners were familiar with the development process, the existing traceability and the goals of the project they reported about. For each project, we first modeled a traceability strategy based on the gathered information. Second, we examined and modeled the applied software engineering processes of each project. Thereby, we focused on executed tasks, involved actors, and pursued goals. Finally, we analyzed the quality and suitability of a projects traceability strategy. We report common problems across the analyzed traceability strategies and their possible causes.

Our paper is organized as follows. Section II reviews related work in the area of requirements traceability. In Section III we outline our research motivation and derive research questions. These questions were used to plan and conduct the interview study described in Section IV. In Section V we describe the analysis procedure step by step and include detailed results from one case. In Section VI we show and discuss results for all cases. Section VII discusses possible threats to the validity of our work and how we mitigated them. Finally, Section VIII concludes our work and outlines future research directions.

II. Related Work

In 1994, Gotel and Finkelstein [1] conducted an extensive study on the so-called traceability problem. The study involved around one hundred software development practitioners, holding a variety of positions within a large organization, with experience ranging between 0.75 and 30 years on a variety of project types. They identified various reasons for that problem, such as lack of training and guidance in traceability practice, failure to follow standard practices, undefined traceability roles, lack of coordination and cooperation between people responsible for different artifacts, and inadequate information about how people contributed to traceability data. Arkley and Riddle [4] identified a lack of motivation for requirements

traceability due to the absence of direct benefits; and a lack of understanding for how to employ traceability. Several studies investigated the negative impact of inadequate traceability on software development. Researchers found that wrong granularity can lead to over-complex or inadequate traceability graphs, and thereby leads to project over-runs or software failures [3], [5]. Ramesh et al. emphasized on the high costs of creating and maintaining traceability, which can only be compensated by higher quality and reduced overall costs if traceability is applied purposefully [6]. Capturing software development activities to automatically generate traces may reduce trace capturing effort [7]. Though, neglecting traceability completely or capturing traces in an unstructured manner to reduce costs will lead to reduced system quality, expensive iterations of defect corrections, and increased project costs [8], [9].

Several authors conducted empirical research on requirements traceability and argue the need for planned traceability and defined traceability strategies. Gotel and Finkelstein argue that knowledge about stakeholders that contributed to traced artifacts helps improving traceability [10]. Ramesh [11] identified two general groups of traceability users, which he refers to as low-end and high-end traceability users. While low-ends users rely on simple dependencies among requirements, high-end users leverage much more sophisticated traceability schemes. Ramesh and Jarke [12] conducted a large practitioner and tool study on traceability. They pointed out that traceability links should be strongly typed in order to avoid semantic misinterpretations. As a result, the authors proposed a traceability meta-model and reference models as guidance for practitioners. We advocated the use of a traceability information model as a necessary condition to employ traceability [13]. Arkley and Riddle [14] conducted a case study on a software project, which successfully leveraged traceability. They concluded that the success of the observed traceability system was mainly influenced by two facts: (i) general traceability needs were examined to support project participants in their tasks (ii) the traceability information model was systematically tailored to the identified needs.

Dömges and Pohl [9] propose a more holistic approach to employ project-specific traceability. Rather than reducing traceability to the definition of permitted artifacts and link types, the definition of project-specific trace strategies is advocated. Thereby, trace capture and usage strategies define what data should be captured and used, in which situation, by whom, and how. Additionally, the authors developed the framework PRIME-RT [15], which provides integrated traceability guidance by automatically reminding, enforcing, and controlling a project-specific traceability strategy. The authors reported that the application of this framework in prototypical experiments lead to better traceability and higher product quality. Though, PRIME-RT supports the application of traceability strategies, the authors did not discuss what is required to define an adequate traceability strategy, which in turn is a necessary to successfully employ the strategy.

In a recent publication [16], we reported on traceability problems that were observed by members of the U.S. Food and Drug Administration (FDA) who systematically evaluate traceability documentation for FDA medical device approval as their daily business. It turned out to be a major problem in current traceability practice that the definition of project specific traceability strategies is either lacking or inappropriate for the investigated project.

III. Research Motivation and Question

As outlined in Section II, different traceability problems are well studied and empirically grounded. Extensive research effort was taken to address the various facets of the traceability problem. Beyond that, commercial CASE tool vendors improved tool support for traceability. However, even most recent studies from the field of traceability practice report about practitioners having problems with properly defining and employing project specific traceability [3], [16].

Given this discrepancy, we decided to conduct an in-depth study on applied traceability strategies in practice. Our research was motivated by the following research questions:

RQ-1 *What traceability strategy do practitioners apply in their development project and is this strategy explicitly defined?*
RQ-2 *Do practitioners create usable traceability?*
RQ-3 *Do practical applied traceability strategies support all project-specific traceability needs?*

The traceability strategy concept is rather abstract. Thus, we decomposed it into concrete traceability concepts, namely, traces between artifacts (what), necessary operations to capture and use traces (how), reasons for leveraging traces (why), and the involved people (who). We derived the following sub-research questions from RQ-1:

RQ-1.1 *What traces between which artifacts do practitioners capture?*
RQ-1.2 *Who is using these traces?*
RQ-1.3 *How do they use traces?*
RQ-1.4 *Why do practitioners use particular traces?*

IV. Study

We applied a qualitative research method for answering our research questions. We choose that method for the following reasons: first, defining a traceability strategy is a complex and multi-faceted challenge. Thus, it would be difficult to define specific context variables. Second, with the chosen approach we were closer to the studied software projects and its participants. So, we gained an understanding of the mechanics behind the observed phenomena and avoided misinterpretations during the analysis.

Another benefit of qualitative research is that it generates rich data that can answer slightly varying research interests in parallel. We used that possibility and gathered data for our main research questions discussed in this paper. We also conducted a small study on inter-organizational traceability [17]. We used different questionnaires for both studies, generating the required data per topic. Both studies involved the

same cases. Participants were not informed about our varying research interests in order to avoid bias.

A. Sampling Cases

We assembled a list of potential companies from the membership list of the association of friends of the Technical University Ilmenau. This list was extended by contacts we made at a practitioners forum on requirements engineering. We considered every company in the resulting list of 85 companies as a potential case for our study. In order to prioritize this list, we collected general information about each company and identified contact persons from the internet. We then developed a case sampling strategy in order to select the most suitable companies and informants for our study. Following the framework of Curtis et al. [18], we defined and used the following sampling criteria:

- How relevant are general case characteristics?
- What is the case potential to generate rich information?
- How generalizable are findings from this particular case?
- What resources are required to study this case?
- Does any ethical issues force us to exclude this case?

After prioritizing the list of potential cases, the contact persons of highest prioritized cases were contacted in order to arrange an interview. Provided that the sampled company agreed, we conducted either one or multiple interviews with key informants who are familiar with the company's software development process and its traceability practice. Every informant was interviewed in an individual interview session to avoid influences between informants.

B. Data Collection

We decided to interview the practitioners face-to-face within their natural working environment. To understand practitioner's perceptions, we especially considered contextual and relational information as important. With every single informant, we conducted an in-depth interview lasting for three to six hours. Additionally, we decided to employ semi-structured interviewing to guarantee that our investigations were guided by our research questions, while keeping the flexibility to react on unforeseen informants responses and to explore unexpected phenomena. Our questionnaire consisted three parts:

1) **General information about informant and working environment:** we collected background information about the informant and his or her working environment as outlined in Table I. We asked the informant to focus on a particular software development project for the remaining part of the interview. We requested that the selected project was representative for the company's software development practice and that she/he was currently involved in it or had recently finished that project (more details on validity in Section VII).
2) **Application of requirements traceability:** we asked the informant about the reasons for applying requirements traceability in her/his concrete project. We collected detailed information on how and why every single project participant used traceability. For this purpose, we recorded all relevant aspects of each traceability usage scenario, such as actors, trace paths, artifacts, tools, tasks, and intention.
3) **Software development process:** we asked for important software process elements in the reported project, such as activities, tasks, actors, stakeholders, goals, artifacts, and tools. Thereby, we aimed to get a holistic view on the software process from beginning to the end.

To ensure focused and high quality results, we conducted a pilot interview before running the actually study. Therefore, we selected a company in close proximity and conducted a three hours lasting interview. In result, we produced interview minutes and field notes. We analyzed the interview minutes in order to reveal and eliminate conceptual weaknesses from the questionnaire. We further conducted a retrospective review of our field notes to improve our interview tactics. Thereby, we realized the necessity to approach certain topics differently in order to avoid unwittingly influencing the informant. The actual interviews were then conducted with 20 informants from 17 different companies. All interviews were recorded in writing by a designated minute taker.

C. Data Demographics

In our study, a case refers to a single software development project in a distinct company (see Table I). Similar to our previous study on traceability practices [3], we observed different key drivers, why traceability is applied in software projects. Informants in six cases reported that they are obligated to provide traceability by *regulations*. Further eight cases stated that their *customer* explicitly demand for traceability. The remaining three cases stated that traceability is used on a voluntary basis by *enthusiasts* to improve product quality.

Our study contains *small* projects (less than 5 project participants), *medium* projects (5 to 10 project participants), *large* projects (10 to 100 project participants), and *huge* projects (more than 100 project participants). These projects were run by *small* (less than 100 employees), *medium* (100 to 1,000 employees), *large* (1,001 to 10,000 employees), and *huge* (more than 10,000 employees) companies. While small and medium companies mainly run small projects, large and huge companies mainly run large and huge projects.

The studied companies are active in various domains (Avionic, Finance, Insurance, Logistics, Retail, E-Commerce, Security, and Transportation) and produce different offerings (Software Product, Hardware Product, Software Development Services). The majority of informants were either *mid-level professionals* (5 to 10 years of work experience) or *senior-level professionals* (more than 10 years of work experience). Only one interviewed informant was a *entry-level professional* (practitioner with less than 5 years of work experience). To mitigate the risk of gathering imprecise information from inexperienced informants, we never interviewed solely entry-level professionals for one particular case.

D. Data Analysis

We analyzed the interview results in a two phase process.

TABLE I
CHARACTERISTICS OF INTERVIEWED INFORMANTS, THEIR PROJECTS, AND COMPANIES

Traceability Driver	Project Members	Company Employees	Case ID	Domain	Offering	Informant ID	Informant's Role	Informant's Experience [yr]
Regulations	5..9	$> 10,000$	10	Insurance	Service	10.1	Process Manager	10..20
						10.2	Release Manager	5..10
	10..100	$1,001..10,000$	9	IT Security	SW Product	9.1	Development Lead	10..20
		$> 10,000$	14	Avionic	HW Product	14.1	Tester	10..20
		< 100	16	Requirements Tool	SW Product	16.1	Development Lead	10..20
	> 100	$1,001..10,000$	3	Finance	Service	3.1	Project Manager	10..20
		$100..1,000$	17	Telecommunication	HW Product	17.1	Process Manager	10..20
Customer	< 5	< 100	6	Robotic	HW Product	6.1	Development Lead	5..10
		$100..1,000$	8	Finance	SW Product	8.1	Project Manager	> 20
	5..9	< 100	2	Insurance	Service	2.1	Project Manager	10..20
		$100..1,000$	7	Finance	Service	7.1	Specification Manager	5..10
			13	Finance	Service	13.1	Compliance Manager	> 20
		$> 10,000$	12	Retail	Service	12.1	Portfolio Manager	5..10
						12.2	Test Manager	10..20
	> 100	$100..1,000$	5	E-Commerce	SW Product	5.1	Development Head	10..20
		$> 10,000$	15	Logistic	Service	15.1	Business Analyst	> 20
Enthusiasm	< 5	< 100	1	Public Service	Service	1.1	Developer	< 5
						1.2	Project Manager	10..20
			4	Retail	SW Product	4.1	Development Lead	5..10
	5..9	< 100	11	Insurance	Service	11.1	Development Lead	10..20

First, we applied qualitative content analysis [19] to systematically extract relevant data from the interview minutes. Our research questions (see Section III) and the interview questionnaire served as qualitative description model. From this model, we derived a system of codes for all three interview parts. These codes were used to classify all written interview minutes and field notes with the qualitative analysis tool[1].

Second, we applied a traceability strategy assessment procedure, which we also derived from our research questions (see Figure 1):

Step 1 Identify and assess existing trace paths [RQ-1.1]: we used traceable artifacts and traces and recovered a conceptual traceability information model per project (what). This conceptual traceability model and the defined trace paths were then assessed for ambiguity, ephemerality, and redundancy.

Step 2 Identify traceability usage goals [RQ-1.2, RQ-1.3, RQ-1.4]: we used information about the purposes of traceability per project and derived a conceptual goal model of traceability usage (who, why, how).

Step 3 Analyze the existing software development process: we used activities, tasks, actors, stakeholders, goals, artifacts and derived a conceptual goal model of the software process.

Step 4 Assess suitability of traceability usage goals [RQ-2, RQ-3]: we assessed the derived traceability usage goals against the derived software process goals and examined whether or not all software process goals of all involved actors were adequately addressed by traceability usage goals. We identified missing and superfluous traceability goals and derived an appropriate traceability usage goal model.

Step 5 Assess suitability of existing trace paths [RQ-2, RQ-3]: we assessed the recovered traceability information model against the appropriate traceability usage goal model (Step 4) for suitability. We identified missing and superfluous traces.

[1]MAXQDA10 – http://www.maxqda.com

V. ASSESSMENT PROCEDURE AND ILLUSTRATING EXAMPLE

In this section we demonstrate the application of our assessment procedure. We randomly selected Case 5 and use it as a running example and a detailed explanation of our analysis procedure within this section. The remaining cases were analyzed and assessed following the same procedure. Discovered results and derived findings are discussed in Section VI.

1) Step 1: Identify and Assess Existing Trace Paths: Due to the fact that only a single project of our study had explicitly defined a traceability information model, we had to detect traceable artifacts and traces between those artifacts in the interview minutes, and derived a conceptual traceability information model. Figure 2 outlines the conceptual traceability information model of Case 5. In Case 5, *Feature Wishes* from different stakeholders are collected through different

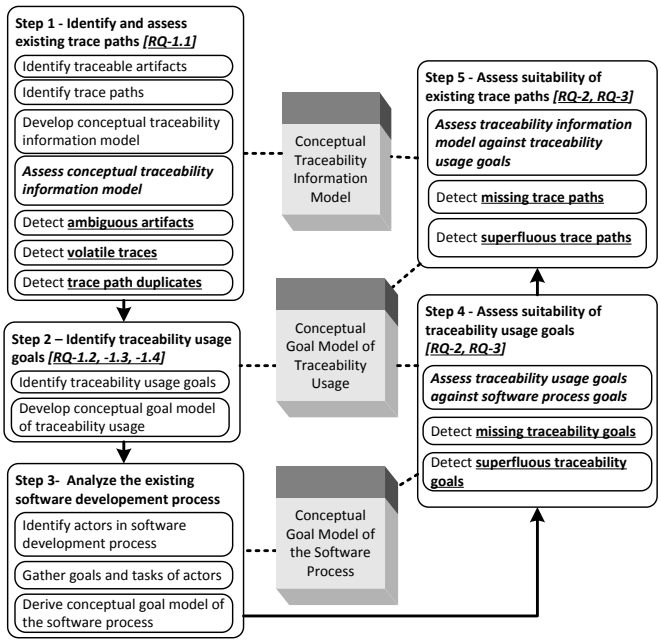

Fig. 1. Derived traceability strategy assement procedure

input channels such as business partners, customers, system integrators, platform hosters, and competitor analysis. These wishes are then consolidated into *Feature Requests*. For every software release iteration, the development head creates a *Release Plan* that specifies what features are going to be implemented within a *Release*. Also, the *Release* contains a detailed break-down structure of work packages and related tasks. Tasks are linked to produce results of every task.

Based on a previous study on traceability problems [16] that were observed by members of the U.S. Food and Drug Administration (FDA) who systematically evaluate traceability documentation for FDA medical device approval as their daily business, we defined the following problem categories and corresponding assessment methods to systematically evaluate the conceptual traceability information model.

a) Artifact Ambiguity Problem: A traced artifact cannot uniquely be identified within the pool of all existing artifacts. *Assessment:* Assess whether or not each artifact is identified by a unique ID. *Result:* Artifacts are classified as *ambiguous* or *non-ambiguous*. *Example:* In Case 5, structural design models are specified as Wiki pages. Structural design artifacts such as model, component, and class are not uniquely named nor do they have unique identifier within the Wiki page. Thus, the artifacts *Structural Design* and *Behavioral Design* are defined as ambiguous artifacts (see Problem #1 in Figure 2).

b) Trace Path Ephemerality Problem: The long-term existence of a traceability path cannot be ensured. *Assessment:* Assess whether or not all traced artifacts and traces are persistently stored. *Result:* Traces are classified as *persistent* or *volatile*. *Example:* In Case 5, the head of development regularly creates a *Release Plan* within a dedicated project management tool. Then, he manually transfers that *Release Plan* into the process centered application life-cycle management (ALM) tool in which it becomes a *Release* artifact. Though, implicit traces through identical naming are existing, these traces cannot be actively retrieved and used. Thus, we consider the trace between the *Release Plan* of the project management tool and the *Release* definition within the process ALM tool as *volatile* (see Problem #2 in Figure 2).

c) Trace Path Redundancy Problem: Redundant traceability paths are defined. *Assessment:* Search for alternative trace paths connecting the same two artifacts in the traceability information model and having the same semantics. *Result:* A set of redundant traceability paths that should be eliminated from the traceability information model. *Example:* This problem was not present in Case 5.

A. Step 2: Identify Traceability Usage Goals

We extracted the following information from our interview minutes: who is using the traces of the conceptual traceability information model *[RQ-1.2]*, why is she or he using those traces *[RQ-1.4]*, and how is she or he using those traces *[RQ-1.3]*. As a result of this analysis, we identified a conceptual goal model of traceability usage per case. For Case 5 this model is shown in Figure 3.

B. Step 3: Analyze the Existing Development Process

While the current traceability strategy of a project was analyzed as described in the two previous steps, these steps did not provide information about the appropriateness of the traceability strategy [*RQ-2*] nor whether or not all project-specific traceability needs were satisfied for a project [*RQ-3*]. A necessary condition for addressing these research questions is that all traceability needs within a project are understood and formalized. To achieve that, we analyzed per project the software development process from the beginning to the end. We extracted a goal model for each software process that contains all process tasks as well as the corresponding goals of every actor. For Case 5 this model is shown in Figure 4.

C. Step 4: Assess Suitability of Traceability Usage Goals

We assessed the goal model of traceability usage against the goal model of the software process [RQ-2 and RQ-3]. Ideally, every process task that requires the usage of traceability should be addressed by at least one traceability usage goal. In result, two types of problems could emerge. First, a software process task that requires traceability usage is not covered by a traceability usage goal. Second, a process task that requires traceability usage is covered by too many usage goals, and thus, unnecessary effort for capturing traceability is generated.

d) Usage Goal Suitability Problem: A software process task that requires traceability usage is not covered by a traceability usage goal or vice versa. *Assessment:* Search for software process tasks that require traceability usage and are not covered by any goal within the conceptual goal model of traceability usage or vice versa. *Example:* The solution architect in Case 5 is responsible for verifying whether or not the design is meeting the requirements specification.

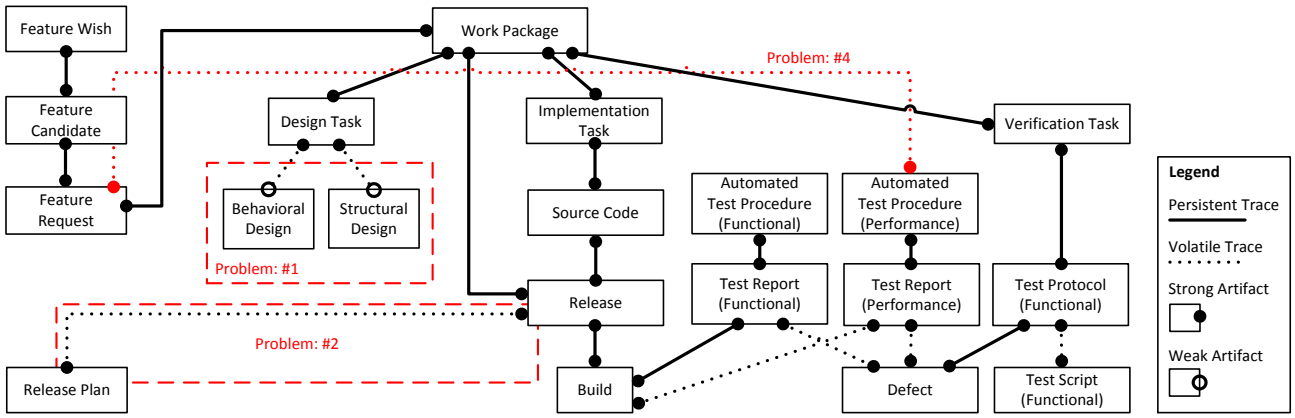

Fig. 2. Recovered conceptual traceability information model for Case 5

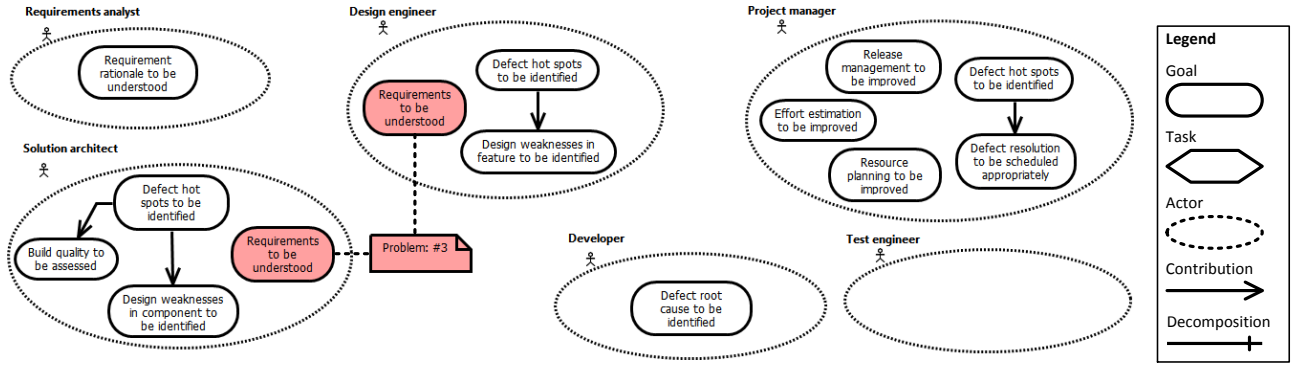

Fig. 3. Derived traceability usage goal model for Case 5

Therefore, she or he must understand the rationale of related requirements, which can be achieved by tracing back from *Feature Request* to related *Feature Wishes*. However, the project's traceability strategy did not contain a usage goal "Requirements to be understood". This usage goal is missing in the conceptual goal model of traceability usage (see Problem #3 in Figures 3 and 4).

D. Step 5: Assess Suitability of Existing Trace Paths

We assessed the conceptual traceability information model against an ideal goal model containing all of traceability usage. This ideal usage goal model was derived based on each project's development process in the previous step. Each task in this usage goal model requires the existence of one or more particular trace paths that may or may not be present in the project. Accordingly, we identified the following problem and applied the related assessment method.

e) Trace Path Suitability Problem: A specific trace path is missing and prevents the execution of a required traceability usage task and the achievement of the related traceability usage goal. Alternatively, a traceability path is superfluous, because none of the identified traceability usage goals requires its existence. *Assessment:* Search for traceability usage tasks that require a specific trace path that is not defined within the conceptual traceability information model. Additionally, search for traceability paths that are not required by any traceability usage goal. *Example:* The performance tester in Case 5 conduct performance tests on a regular basis by running the *Automated Test Procedure (Performance)*. Due to the missing trace path between *Automated Test Procedure (Performance)* and *Feature Request*, performance test results cannot be related to explicit software performance requirements missing the chance to support the tester with this difficult task (see Problem #4 in Figure 2).

VI. Results and Discussion

In this section, we report on the overall findings we extracted from our study. To generate the following data, we analyzed the interview minutes from each case as exemplary described in Section V. Table II summarizes the overall results per problem category. Every problem category consists of one or more classes.

A. Artifact Ambiguity Problem

We considered a traced artifact that cannot uniquely be identified within the pool of all existing artifacts as ambiguous artifact. The first row of Table II outlines the absolute and relative occurrence of ambiguous artifacts per case.

Reasons: Artifact types were conceptually considered but did not exist in digital form. We found especially artifacts

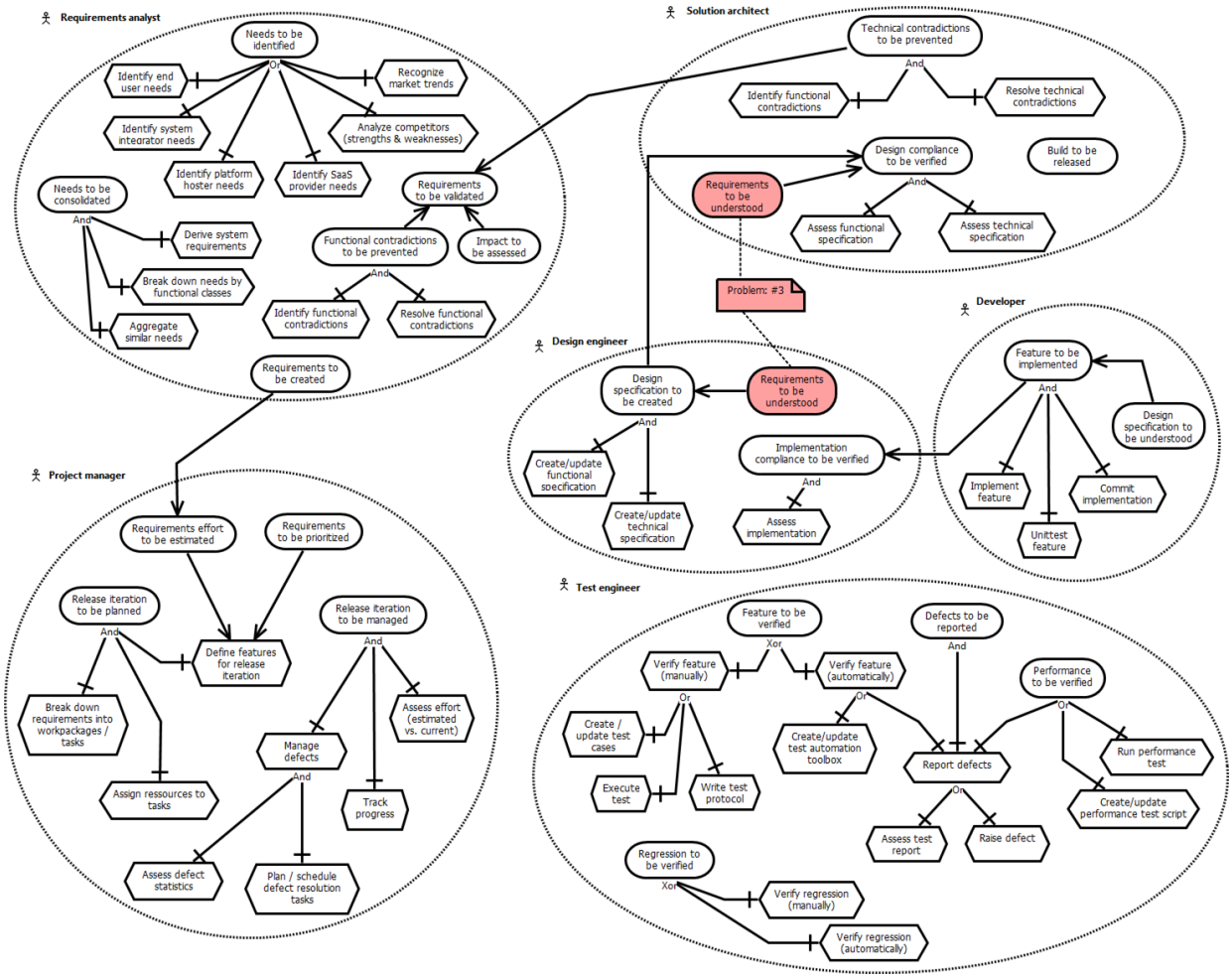

Fig. 4. Derived goal model for the software development process of Case 5

supporting pre-requirements traceability as being problematic. Further, tools required to create or modify artifacts did not provide sufficient support for the generation or storage of unique artifact identifiers. Several years ago, Aizenbud-Reshef et al. [20] referred to the insufficiency of software tools to uniquely identify artifacts across time and space. The authors predicted that future tools will create and maintain globally unique identifiers for artifacts avoiding the ambiguity problem. The results of our study show that their prediction cannot be confirmed yet. Ascuncion et al. [21] observed similar issues concerning the ambiguity problem in a recently conducted industrial case study on end-to-end software traceability.

Complex environments with multiple monolithic tools were lacking cross-tool mechanism to ensure uniqueness of artifact identifiers across the entire platform.

Influence factor traceability driver: The more strict (Regulations) the traceability driver was, the higher was the percentage of non-ambiguous artifacts. Informants from strictly regulated projects stated that traceability is a strategic project goal. Budget for well suited traceability tools is available in such projects. Projects with less strict traceability driver (Enthusiasm) reported that existing tools must be used no matter how viable they are for traceability, often causing the artifact ambiguity problem.

Influence factor project size: Large and huge projects tend to have higher percentage of non-ambiguous artifacts. Due to the economy of scale, expensive traceability tools can only be funded by project budgets of large projects.

B. Trace Path Ephemerality Problem

We considered a trace path as volatile if the traces were not persistently stored, and thus, the long-term existence and usage of traces could not be ensured.

Reasons: We identified the following reasons, why volatile trace paths existed in the conceptual traceability information model. (i) Trace paths were not persistently stored. (ii) Trace paths between artifacts of multiple tools were not stored in a single repository. (iii) Trace paths were conceptually modeled, but no real path created. (iv) Trace paths were defined between artifacts where at least one related artifact was ambiguous.

Influence factor traceability driver: Contrary to our expectations, we found that cases with a more strict traceability driver (Regulations) tend to have more volatile traces than cases with a less strict traceability driver (Enthusiasm). Informants from

TABLE II
APPEARANCE OF ASSESSED PROBLEMS ACROSS THE 17 STUDIED CASES

Problem	**Case ID**																
Class	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Artifact ambiguity problem																	
Ambiguous artifacts	9	2	2	9	2	3	6	3	4	2	2	8	10	5	7	0	9
	62%	17%	15%	75%	11%	43%	33%	25%	27%	12%	29%	50%	67%	36%	37%	0%	50%
Trace path ephemerality problem																	
Volatile traces	5	4	7	8	7	4	7	3	8	6	3	7	9	7	11	0	10
	71%	29%	50%	73%	32%	76%	39%	23%	40%	32%	60%	50%	75%	58%	58%	0%	56%
Trace path redundancy problem																	
Trace path duplicates	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Usage goal suitability problem																	
Missing goals																	
Requirements Analyst	2	1	3	0	4	1	0	0	0	0	2	2	1	0	0	2	2
Solution Architect	0	0	1	1	3	0	0	0	0	0	0	0	1	0	1	0	0
Design Engineer	1	0	0	0	3	0	2	0	0	2	0	0	2	2	0	0	1
Project Manager	3	5	2	3	1	2	1	3	1	0	2	0	2	0	0	3	3
Developer	1	1	0	1	1	0	0	0	0	0	0	1	1	0	2	0	0
Test Engineer	4	3	5	3	5	1	3	3	2	1	3	2	3	2	3	4	2
Superfluous goals	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Trace path suitability problem																	
Missing trace paths	5	6	6	9	9	7	4	5	2	1	3	6	12	3	9	4	7
Superfluous trace paths	0	2	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0

highly regulated projects mentioned that the completeness of traceability is of highest importance for audits. Due to time pressure, ad-hoc traces were created to satisfy auditors. Those ad-hoc traces were predominantly volatile.

Influence factor project type: We identified a strong tendency that projects in product-oriented companies had considerably less volatile traces than those in service-oriented companies. We observed that product-oriented companies applied a more homogeneous development process with a single tool platform or a highly integrated tool chain. Additionally, product-oriented companies considered the time after project completion as equally important as the project itself. Thus, they had a strong interest in persistent traces. Service-oriented companies had typically a more heterogeneous tool landscape as different customers forced them to use different tools. Informants of service-oriented companies stated that persistent traces are less important for them since volatile traces are cheap and sufficient to deliver the project. Investments in expensive persistent traces were not beneficial for short term project, and thus, were considered as cost drivers.

C. Trace Path Redundancy Problem

We considered trace paths as redundant if similar trace paths exist that carried the same semantics. We observed this problem only in one particular case (Case 2), where the project manager defined a rather complex traceability information model. This traceability information model contained three redundant trace paths.

D. Usage Goal Suitability Problem

We identified missing traceability usage goals for all cases of our study (see Table II). That means that all studied projects struggled with defining suitable traceability usage goals.

Reason: Several informants stated that they consider the definition of a suitable traceability usage goal model as extremely challenging due to the complexity of the software process and the multiplicity of actors. They further stated that guidance is required on defining suitable traceability usage goals. In [22] we discovered that a great number of different traceability usage scenarios are frequently used by practitioners in software development projects. Similarly, Ramesh et al. [6] identified several traceability usage scenarios in an industrial case study. When defining a traceability strategy, all traceability usage scenarios and related traceability usage goals could potentially be relevant and must be considered.

Influence factor project role/activity type: We observed that usage goals were missing mainly for three perspectives of the development process, namely *Requirements Analyst*, *Project Manager*, and *Test Engineer*. We further analyzed our results for a possible reason for this phenomena. We found that traceability usage goals are mainly related to four types of tasks in the software development process: planning, monitoring, validation, and verification. The above mentioned perspectives *Requirements Analyst*, *Project Manager*, and *Test Engineer* were most often concerned with those planning, monitoring, validation, and verification tasks.

Influence factor project context: We found that cases gov-

erned by regulation or customer were defining specific usage goals to address the information needs of the regulatory body or customer. Because regulator or customer had documented the information need in advance via contract, or regulatory code, these usage goals were almost complete across the cases. Ramesh [11] made a similar observation and identified that the recognition of traceability problems and the formulation of traceability goals are influenced by the project context.

E. Trace Path Suitability Problem

We observed that the amount of missing trace paths was related to the amount of missing traceability usage goals. All cases were lacking trace paths. More than half of the cases were lacking a considerably high amount of trace paths.

Reason: Except for a single case, no traceability information model was explicitly defined in advance of our study. Similarly to the traceability usage goals, informants stated that they consider the definition of a traceability information model as to challenging due to diversity and complexity of the software development process. Informants also stated that tools are lacking, which provide capabilities to define, monitor and enforce the traceability information model and that are fully integrated with the software development tool chain. In a previous study [3], we noticed that practitioners do not gain enough value from the definition of a information model.

F. Summary

We conclude that all cases were lacking traceability usage goals and required trace paths. In some cases, we even observed a considerable amount of missing traceability usage goals or required trace paths. Nevertheless, the majority of the studied cases failed to define and apply a suitable project-specific traceability strategy. We believe that the research must proceed on this topic as traceability practitioners are currently far away from applying suitable traceability strategies.

VII. Threats to Validity

When planning and conducting our study we carefully considered validity concerns. This section discusses how we mitigated threats to the validity.

A. External Validity

Due to their nature, interview studies cannot be replicated as identical interview circumstances cannot be recreated. Qualitative studies are primarily concerned with describing and understanding existing phenomena. We described such observed phenomena from our interviews (see Section VI). The fact that our cases diverge across multiple domains, locations, and sizes contributes to the applicability of our findings. However, we are aware of the fact that this kind of study is not generalizable.

B. Internal Validity

The instrumentation threat was addressed by applying qualitative content analysis [19], which must be guided by theory from the beginning. We systematically derived a qualitative description model for our study based on the defined research questions. Activities of our study, like creating the questionnaire, conducting the interviews, and extracting the findings were all guided by this qualitative description model as described in Section IV.

C. Construct Validity

We mitigated the threat of case relevance to validly address the research questions by carefully analyzing every studied case in advance. Therefore, we collected general information about the company and the potential informant from the internet. Previous to the actual interview, we gathered further information from the informant via telephone. We especially emphasized that the potential informant had extensive experience in software development projects and is capable to provide insights of the whole process. We carefully analyzed all gathered information and assessed this information against our qualitative description model. Due to the fact that our qualitative description model was derived from our research questions, we are confident that our studied cases are relevant to address our research questions (see also Section III).

In our study, we constructed a traceability usage goal model from the analyzed software development process as described in Section V. We opposed this model to the traceability usage goal model, which we directly created from the statements of our informants. As outlined in Table II, we identified several problems across the 17 studied cases due to missing traceability usage goals. Although, we were able to identify many missing goals, we do not have empirical evidence that we identified all missing traceability usage goals. To mitigate this threat, we carefully assessed the results of our preliminary prototypical interview. We plan a continuing study to provide more empirical evidence regarding this issue.

D. Conclusion Validity

We employed a preliminary prototypical interview under realistic conditions to improve our questionnaire and our questioning technique. Thereby we emphasized on eliminating influencing information from questions or questioning behavior. All interviews of our study were conducted with one informant in a single session without break. We offered no room for distractions and interruptions during the interview in order to avoid influences on subjects' answers.

VIII. Conclusions and Future Work

In this paper we focused on the discrepancy between research and practice in traceability strategy definition. While researchers argue that effective traceability requires an upfront explicitly defined traceability strategy, previous studies showed that practitioners rarely follow such explicit strategies. We set up an interview study in order to assess that phenomenon and the effectiveness of traceability strategies actually applied in practice. As precondition for our work, we derived an analysis procedure and five possible problem types that we assessed. Apart from the actual results of the study, this assessment procedure and the problems types we derived are an important contribution for practitioners and other researchers who want to assess traceability strategies by themselves.

All studied projects contained at least several of the assessed problems in multiple instances. One project (Case 16) showed fewer problems than the remaining ones. Specific about these was the application of an integrated tool-chain. The importance of proper instrumentation for successful requirements traceability is not a new finding, but our results emphasize it again and show its importance. Apart from this case, we found ambiguous artifacts across all projects and traces that were often far from being ready-to-use. Furthermore, we found for most projects a considerable mismatch between the goals of the applied development process, the reported traceability goals, and the actual existing traceability. All projects missed at least some required traceability paths. Making traceability not or not fully suitable for the individual project.

These findings suggest that an upfront-defined traceability strategy is indeed required. However, we showed in this paper that it is a complex task to determine all suitable trace paths for a project. The decision for or against trace paths between artifacts requires a detailed understanding of the projects engineering process and goals. We conclude that traceability strategies should be defined and assessed in a goal-driven procedure. Future work will focus on developing techniques that support practitioners in that procedure. Furthermore, we are working on a continuing empirical study with multiple feedback iterations to improve the proposed methodology for assessing the quality of existing traceability usage strategies in software development projects.

ACKNOWLEDGMENT

The authors would like to thank all practitioners participating in the interview study. We are supported by the German Research Foundation (DFG): Ph49/8-1 and the German Ministry of Education and Research (BMBF): Grant No. 16V0116.

REFERENCES

[1] O. C. Z. Gotel and A. C. W. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings 1st International Conference on Requirements Engineering*. IEEE, 1994, pp. 94–101.

[2] O. Gotel, J. Cleland-Huang, J. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, J. Maletic, and P. Mäder, "Traceability fundamentals," in *Software and Systems Traceability*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer London, 2012, pp. 3–22.

[3] P. Mäder, O. Gotel, and I. Philippow, "Motivation matters in the traceability trenches," in *Proceedings 17th International Conference on Requirements Engineering*. IEEE, 2009, pp. 143–148.

[4] P. Arkley and S. Riddle, "Overcoming the traceability benefit problem," in *Proceedings 13th International Conference on Requirements Engineering*. IEEE, 2005, pp. 385–389.

[5] D. Leffingwell, "Calculating your return on investment from more effective requirements management," Rational, Tech. Rep., 1997. [Online]. Available: http://www.ibm.com/developerworks/rational/library/347.html

[6] B. Ramesh, T. Powers, C. Stubbs, and M. Edwards, "Implementing requirements traceability: a case study," in *Proceedings 2nd International Symposium on Requirements Engineering*. IEEE, 1995, pp. 89–95.

[7] I. Omoronyia, G. Sindre, and T. Stålhane, "Exploring a bayesian and linear approach to requirements traceability," *Information & Software Technology*, vol. 53, no. 8, pp. 851–871, 2011.

[8] K. Pohl, "PRO-ART: Enabling requirements pre-traceability," in *Proceedings 2nd International Conference on Requirements Engineering*. IEEE, 1996, pp. 76–84.

[9] R. Dömges and K. Pohl, "Adapting traceability environments to project-specific needs," *Communications of the ACM*, vol. 41, pp. 54–62, 1998.

[10] O. Gotel and A. Finkelstein, "Extended requirements traceability: Results of an industrial case study," in *Proceedings 3rd IEEE International Symposium on Requirements Engineering*. IEEE, 1997, pp. 169–178.

[11] B. Ramesh, "Factors influencing requirements traceability practice," *Communications of the ACM*, vol. Volume 41 Issue 12, no. 12, pp. 37 – 44, 1998.

[12] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE Transactions on Software Engineering*, vol. 27, no. 1, pp. 58–93, 2001.

[13] P. Mäder, O. Gotel, and I. Philippow, "Getting back to basics: Promoting the use of a traceability information model in practice," in *Proceedings of the ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, 2009, pp. 21 –25.

[14] P. Arkley, S. Riddle, and T. Brookes, "Tailoring traceability information to business needs," in *Proceedings 14th International Conference on Requirements Engineering*. IEEE, 2006, pp. 239–244.

[15] K. Pohl, K. Weidenhaupt, R. Dömges, P. Haumer, M. Jarke, and R. Klamma, "PRIME – toward process-integrated modeling environments," *ACM Transactions on Software Engineering and Methodology*, vol. 8, no. 4, pp. 343–410, 1999.

[16] P. Mäder, P. Jones, Y. Zhang, and J. Cleland-Huang, "Strategic traceability for safety critical projects," *IEEE Software*, vol. 30, no. 3, pp. 58–66, 2013.

[17] P. Rempel, P. Mäder, T. Kuschke, and I. Philippow, "Requirements traceability across organizational boundaries - a survey and taxonomy," in *Requirements Engineering: Foundation for Software Quality*, ser. Lecture Notes in Computer Science, J. Doerr and A. Opdahl, Eds. Springer, 2013, vol. 7830, pp. 125–140.

[18] S. Curtis, W. Gesler, G. Smith, and S. Washburn, "Approaches to sampling and case selection in qualitative research: examples in the geography of health," *Social Science & Medicine*, vol. 50, no. 7, pp. 1001–1014, 2000.

[19] P. Mayring, "Qualitative content analysis," in *Forum: Qualitative Social Research*, vol. 1, no. 2, 2000. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:0114-fqs0002204

[20] N. Aizenbud-Reshef, B. T. Nolan, J. Rubin, and Y. Shaham-Gafni, "Model traceability," *IBM Systems Journal*, vol. 45, no. 3, pp. 515–526, 2006.

[21] H. U. Asuncion, F. François, and R. N. Taylor, "An end-to-end industrial software traceability tool," in *Proceedings 6th joint meeting of the European Software Engineering Conference and the International Symposium on Foundations of Software Engineering*. ACM, 2007, pp. 115–124.

[22] 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. Springer, 2013, vol. 7830, pp.