Thesis for The Degree of Doctor of Philosophy

Improving software traceability tools and processes

Salome Honest Maro

Division of Software Engineering
Department of Computer Science & Engineering
University of Gothenburg
Gothenburg, Sweden, 2020.

Improving software traceability tools and processes

Salome Honest Maro

ISBN 978-91-8009-012-4 (PRINT)
ISBN 978-91-8009-013-1 (PDF)

Technical Report No 186D
Department of Computer Science & Engineering
Division of Software Engineering
University of Gothenburg
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2020.

To my family

Abstract

Context: Development of large and complex software leads to a large number of interconnected artifacts such as requirements, design models, code and implementation. Traceability enables understanding and managing these artifacts as they evolve. However, establishing traceability is not trivial. It requires the development organization to design effective traceability strategies and provide tools to support the traceability activities. Establishing traceability in practice faces many challenges such as the amount of effort needed to establish traceability, unclear traceability processes and difficulty in measuring the benefits of traceability.

Objective: The overall objective of this research is to improve traceability processes and tools in software development. In this thesis we started with exploring the state of the art as well as the state of practice of traceability in order to identify persisting challenges and existing solutions. We then propose and implement solutions for four of the identified challenges: manual work of establishing traceability, lack of configurable tools, diverse artifacts and tools, and unclear traceability processes.

Method: To identify existing traceability challenges and solutions, we conducted a systematic tertiary literature review, a multi-vocal literature review, and a case study to understand how these challenges and solutions manifest in practice. To design solutions we took an empirical approach where we used case studies and design science for the different studies.

Results: Our results show that there are still many traceability challenges which are not solved by current solutions in literature due to practical constraints and limitations that exist especially in safety critical domains. To address the challenge of manual work needed to maintain trace links we propose a set of important factors and guidelines for traceability maintenance aimed at traceability tool developers and companies acquiring traceability tools. The feasibility of these factors and guidelines are demonstrated through a prototype implementation. The prototype implementation also shows how to design traceability solutions that are both configurable and support tracing to and from diverse artifacts and tools. To address the challenge of manual work in creating traceability links we have investigated how to improve the trace link vetting process as part of a way to transfer automated techniques of creating trace links to industry. We provide insights and lessons learned on how to improve the trace link vetting process. Lastly the thesis proposes a traceability introduction methodology (TracIMo), which consists of concrete steps that companies can take to design, deploy and evaluate traceability strategies.

Keywords Traceability, Software Traceability

Acknowledgment

I would like to thank my supervisors, Jan-Philipp Steghöfer and Miroslaw Staron. To Jan-Philipp, thanks for the ideas, guidance, discussions, extensive feedback (including code reviews) and always being there to support me. To Miroslaw, thanks for providing great support, feedback and steering me to the right research directions. I would also like to thank my examiner Jan Jonsson for providing positive feedback and being very encouraging throughout this whole PhD journey.

I would also like to thank my previous supervisor Matthias Tichy for the support provided during the time he acted as my supervisor.

Lots of thanks to all my co-authors for providing support and making my research and all the papers possible. Special thanks to Jane Cleland-Huang and Jane Huffman Hayes for the research visit opportunity. I would also like to thank all the Eclipse Capra contributors for making the tool a great success.

Special thanks to all the colleagues in the Software Engineering division, my fellow PhD students, faculty and administrative staff for providing a great and supportive work environment.

Behind every successful person, there is a supporting family and a group of great friends. I would like to thank my parents (Honest and Radegunda Maro) and my sisters (Judy, Rose and Vicky), for their continuous support and encouragement. To my friends, especially Jacky(Boko), Asna and Julitha (Cuzo), thanks for the love and encouragement. To Lucy Lwakatare, thanks for providing great discussions, support and a work environment for me during the pandemic. To my mother in law (Agnes), thanks for spending so much time and helping me take care of my daughter during my work hours. Most important, to Seif Hamad, my love, thanks for your support, encouragement and patience. Lastly, to my daughter Maya, you are just awesome and give me lots of joy and strength in life.

Part of the work reported in this thesis was conducted as part of the AMALTHEA4Public project and the PANORAMA project both funded by ITEA 3.

List of Publications

Appended publications

This thesis is based on the following publications:

[A] S. Maro, J.-P. Steghöfer, M. Staron "Software traceability in the automotive domain: Challenges and solutions."
Journal of Systems and Software 141 (2018): 85-110.

[B] S. Maro, A. Anjorin, R. Wohlrab, J.-P. Steghöfer "Traceability Maintenance: Factors and Guidelines"
31st International Conference on Automated Software Engineering (ASE 2016), Singapore, Singapore, September 3-7, 2016.

[C] S. Maro, J.-P. Steghöfer "Capra: A Configurable and Extendable Traceability Management Tool"
24th International Conference on Requirements Engineering (RE2016), Beijing, China, September 12 - 16, 2016.

[D] S. Maro, J.-P. Steghöfer, J. Hayes, J. Cleland-Huang, M. Staron "Vetting automatically generated trace links: What information is useful to human analysts?."
26th International Requirements Engineering Conference (RE) 2018 Aug 20 (pp. 52-63). IEEE.

[E] S. Maro, E. Sundklev, C-O. Persson, G. Liebel, J.-P. Steghöfer "Impact of Gamification on Trace Link Vetting: A Controlled Experiment"
International Working Conference on Requirements Engineering: Foundation for Software Quality 2019 Mar 18 (pp. 90-105). Springer, Cham.

[F] S. Maro, J.-P. Steghöfer, P. Bozzelli, H. Muccini "TracIMo: A Traceability Introduction Methodology and its Evaluation in an Agile Development Team"
Under revision at the Requirements Engineering Journal.

Other publications

The following publications were published during my PhD studies. However, they are not appended to this thesis, due to contents overlapping that of appended publications or contents not related to the thesis.

[a] S. Maro, J.-P. Steghöfer, A. Anjorin, M. Tichy "On integrating graphical and textual editors for a UML profile based domain specific language: an industrial experience"
International Conference on Software Language Engineering 2015 Oct 26 (pp. 1-12).

[b] R. Wohlrab, J.-P. Steghöfer, E. Knauss, S. Maro, A. Anjorin "Collaborative Traceability Management: Challenges and Opportunities"
24th International Conference on Requirements Engineering (RE 2016), Beijing, China, September 12 - 16, 2016.

[c] M. Trei, S. Maro, J.-P. Steghöfer, T. Peikenkamp "An ISO 26262 Compliant Design Flow and Tool for Automotive Multicore Systems"
17th International Conference on Product Focused Software Process Improvement (PROFES, 2016), Trondheim, Norway, November 22-24, 2016.

[d] S. Maro, M. Staron, J.-P. Steghöfer "Challenges of Establishing Traceability in the Automotive Domain"
9th International Conference on Software Quality (SWQD 2017), Vienna, Austria, January 17-20, 2017.

[e] S. Maro "The Automotive Domain – From Multi-Disciplinarity to Trans-Disciplinarity"
Multidisciplinary Digital Publishing Institute Proceedings 2017 (Vol. 1, No. 3, p. 172).

[f] S. Maro "Addressing Traceability Challenges in the Development of Embedded Systems"
Lic. Phil. Chalmers University of Technology and University of Gothenburg, 117 p, 2017.

[g] M. Mukelabai, D. Nesic, S. Maro ,T. Berger, J.-P. Steghöfer "Tackling combinatorial explosion: a study of industrial needs and practices for analyzing highly configurable systems"
International Conference on Automated Software Engineering 2018 Sep 3 (pp. 155-166).

[h] R. Wohlrab, E. Knauss, J.-P. Steghöfer, S. Maro, A. Anjorin, P. Pelliccione "Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture"
Requirements Engineering Journal. 2018 Nov 1:1-25.

[i] R. Kasauli, R. Wohlrab, E. Knauss, J.-P. Stegöfer, J. Horkoff , S. Maro. "Charting Coordination Needs in Large-Scale Agile Organisationswith Boundary Objects and Methodological Islands"
International Conference on Software and System Processes, ICSSP 2020.

[j] S. Maro, J.-P. Stegöfer, E. Knauss, J. Horkoff, R. Kasauli, R. Wohlrab, J-L. Korsgaard, F. Wartenberg, N.J. Strøm, R. Alexandersson “Managing Traceability Information Models: Not such a simple task after all?” Under revision at IEEE Software

Research Contribution

This section outlines my individual contributions for each of the papers appended in this thesis.

In Paper A, I took the lead in design of the study, performing the systematic literature review and the case study. The first data collection of the multi-vocal literature review was done by my supervisor, but I took part in the screening as well as analysis of the data in the study. I also played a lead role in writing up the paper.

In Paper B, I was responsible for collecting data on traceability management practices and tools from project partners, analysis of the data and conducted the evaluation interviews as well as the analysis of the data. I also wrote the majority of the publication with support from the co-authors.

Paper C, reports on a tool that was developed with me in collaboration with other developers. I was involved in the conceptualization phase and initial implementation of the tool and I am currently the project lead for the Eclipse Project of the tool.

In Paper D, I took the lead in designing the study, conducting the interviews, developing the tool and analyzing the data. I was also lead in writing the paper. The co-authors provided support by giving feedback to the study design, helping in finding study participants as well as reviewing the paper.

Paper E is a result of a master thesis that I conceptualized and designed. It used the tool implemented in Paper D as a basis. I supervised the thesis together with one of the co-authors and participated in writing of the final paper.

In Paper F, I took the lead in the design of the study, conducting interviews, analyzing the results, customizing the tool and deploying the tool in the company. I lead the evaluation as well as the writing of the paper. The co-authors provided support with shaping the study design, support during deployment and provided extensive feedback during paper writing.

Contents

Chapter 1

Introduction

Many products today such as cars, health care devices and airplanes contain software. As software is increasingly being used in complex tasks, its complexity also increases. With this increase in complexity, a large number of artifacts such as requirements, models, code and tests are being produced during development. For instance, the software in a modern car consists of around 100 million lines of code and up to 20,000 pages of requirements [1]. This complexity is also observed in other domains like the telecommunication domain where software in telephone switches contains around 200 million lines of code [2] and about 10,000 requirements [3]. Such complexity makes it difficult to ensure that the delivered software and systems work as expected. This is because these artifacts do not exist in pure isolation but are related to each other and their consistency needs to be ensured during the entire development life cycle. Managing a large number of artifacts is difficult for both developers and other stakeholders involved in software development.

One factor that enables ensuring that software works as expected is the ability to connect and relate different development artifacts, for instance, the ability to connect customer requirements to detailed technical requirements, implementation and tests. This gives the possibility to demonstrate how each requirement has been implemented and tested. Traceability is the ability to relate different development artifacts and is therefore very important as it can be used to reason about the relationships between the different artifacts. Traceability has many other benefits including increasing program comprehension, facilitating impact analysis, facilitating tracking of project progress and supporting change propagation [4–7]. However, for these benefits to be realized, a development organization needs to invest in establishing traceability. This means putting in place processes on how traceability links will be created, maintained and used, and providing tools to support these activities.

Establishing traceability in practice still remains a challenge since in many development organizations, traceability practices are poor, mainly due to lack of well-defined processes and tool support for establishing traceability [8, 9]. The overall goal of this research is therefore to *improve traceability* ***processes*** *and* ***tools*** *in software development.* This goal is broken down into two subgoals:

Goal 1: Identify current traceability challenges and solutions.

Goal 2: Explore and propose solutions for the existing traceability chal-

lenges related to **tools and processes.**

Explicitly, for **Goal 1**, our contribution is a set of challenges and existing solutions for software traceability elicited from a tertiary literature review, a multi-vocal literature review and a case study. For **Goal 2**, the thesis provides a number of contributions:

- A set of important factors and guidelines for traceability maintenance.
- A prototype implementation of how the guidelines can be implemented.
- A set of lessons learned from experiments on how to improve the traceability vetting process (which is the process of validating automatically generated trace links).
- A methodology for introducing traceability that can be applied in companies that want to introduce traceability in their development environment.

The rest of this chapter is structured as follows: First a background on the topic of traceability is given in Sections 1.1, 1.2 and 1.3 where we discuss definitions of traceability, traceability strategies and traceability management tools. The scope of the thesis is described in Section 1.4 and related work in Section 1.5. The methodology for the research and how the threats to validity were mitigated are reported in 1.6 and 1.7 respectively. The contributions of the thesis are given in Section 1.8. Section 1.9 gives a discussion on how the research questions were answered and Section 1.10 outlines our future work. Finally Section 1.11 concludes the chapter.

1.1 Traceability Definition

In literature, there are a number of definitions of traceability. This section gives an overview of the most cited traceability definitions, analyses the definitions and gives the definition of traceability that is used in this thesis. This analysis is summarized in Table 1.1

Gotel et al. [10] give a general definition of traceability as “the potential for traces (a specified triplet of elements comprising: a source artifact, a target artifact and a trace link associating the two artifacts) to be established (created and maintained) and used”. The authors define a trace artifact as a “traceable unit of data”. This definition explicitly mentions the use of trace links implying that links should only be established if they will be used. The shortcoming of this definition is that it is recursive as traceability is defined using the term traces and trace artifact is defined using the word traceable.

In line with the Gotel et al. [10] definition, the Center of Excellence for Software and System Traceability (COEST) [11] defines software traceability as “the ability to interrelate any uniquely identifiable software engineering artifact to any other, maintain required links over time, and use the resulting network to answer questions of both the software product and its development process”. This definition states that the artifacts need to be unique which is a characteristic of a traceable artifact [12] and also that the traces should be maintained over time and used in the development process. However from the definition, it is not clear what the authors mean by “maintain required links".

This may imply that some links that are not required may be created but not maintained. Also the "over time" is very generic and may be interpreted in several ways, e.g., to mean forever or for a certain period of time.

Another definition is provided by Spanoudakis and Zisman [5] who define traceability as "the ability to relate artifacts created during the development of a software system to describe the system from different perspectives and levels of abstraction with each other, the stakeholders that have contributed to the creation of the artifacts, and the rationale that explains the form of the artifacts". Like the previous definition, this one also stresses the use of traceability links and even specifies the kinds of uses; for instance being able to explain the rationale of the artifacts. Making the uses explicit in the definition gives an impression that these are the only uses of traceability while in reality there are many more, such as, facilitating tracking of project progress and reuse of artifacts [4].

Older definitions are given in the IEEE standard glossary of software engineering terminology [13] which gives two definitions of traceability:

[a] "The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; for example, the degree to which the requirements and design of a given software component match". This definition does not mention anything suggesting that these relationships must be useful. Additionally, even though the example given may give an idea of what "product" means in this case, the word "product" may be interpreted as a complete software or system implying a certain level of granularity for the traced artifacts.

[b] "The degree to which each element in a software development product establishes its reason for existing; for example, the degree to which each element in a bubble chart references the requirement that it satisfies". Like the first definition, this one also does not mention that the relationships should be useful. On the other hand, this definition uses the term "element" which could refer to a more fine grained granularity compared to the term "product" in the previous definition.

There are also definitions which are explicitly from a requirements perspective. Gotel & Finkelstein [14] define *requirements traceability* as "the ability to follow the life of a requirement in both forwards and backwards direction (i.e., from its origins through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)." This is the only definition that discusses refinement and iterations of artifacts. This implies not only being able to "follow" a requirement to other artifacts but also being able to follow and track the different versions. However this definition is only focused on requirements. It does not explicitly mention the need to connect intermediate artifacts that may not be related to requirements. Also the definition only informs about being able to follow the life of a requirement but does not mention anything about the use of traceability. A similar definition that is requirements-oriented is provided by Spanoudakis [15] which defines requirements traceability as the "ability to relate requirements specifications with other artifacts created in the development life cycle of a software system".

Source	Trace What?	To what?	Trace when?	Why?
[10]	Development artifact	Development artifact	–	To be used
[11]	uniquely identifiable software engineering artifact	uniquely identifiable software engineering artifact	over time	use the resulting network to answer questions of both the software product and its development process
[5]	Development artifact	Development artifact	–	to describe the system, the stakeholders and the rationale of artifacts
[13]	product of software development process	product of software development process	–	–
[13]	element in a software development process	element in a software development process	–	–
[14]	Requirements	development artifacts	Development life cycle including use in the field	–
[15]	Requirements	development artifacts	Development life cycle	–

Table 1.1: Analysis of Traceability Definitions

Analyzing all these definitions, we can deduce that there are three aspects that are important to be able to define traceability. These are: *artifacts involved, which point in the development and the purpose of the links.* We can reduce this into *what*, *when* and *why* questions (Table 1.1). The question of *how* traceability links are established is orthogonal to all of these and thus not included in this classification.

Based on this analysis, in this thesis, we use the definition of software traceability similar to the one given by COEST [11], but slightly adopted to emphasize software evolution as well.

Software Traceability. *The ability to relate uniquely identifiable software engineering artifacts created and evolved during the development of a software, maintain these relationships throughout the entire development life cycle and use them to facilitate software development activities.*

Software development artifacts in this case include requirements, design models, behavior models, code, test cases, test results and all other artifacts that are related to the system. We stress inclusiveness of development artifacts because in many cases there has been confusion about the scope of traceability where some practitioners seem to think that it is limited to requirements artifacts [16]. We also stress the usefulness of the links because it is possible to create links to arbitrary artifacts but this is not only a waste of time when the links have no purpose, but also creates a lot of noise and may make the useful links difficult to find and use. This definition is similar to the one by

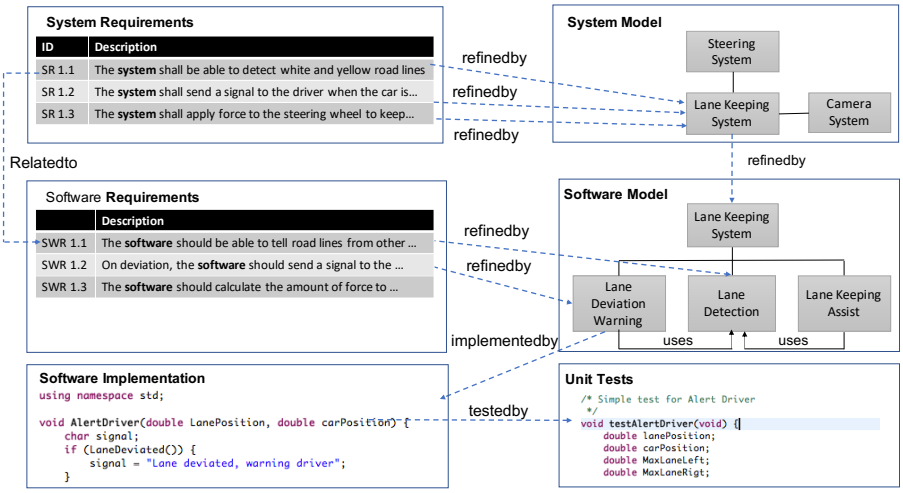


Figure 1.1: A simplified example of artifacts in a lane keeping system and traceability links involved.

COEST [11], but substitutes “over time” with ”entire development life cycle" to make the *when* explicit and includes the word “evolved” to indicate that artifacts can have different versions and these versions should be traceable, a factor which is deemed important in this thesis.

1.1.1 Example

This section uses an example of a lane keeping system from the automotive domain to demonstrate what traceability looks like in practice. A lane keeping system is a system that is used to identify lanes on the road and help the driver keep the car within the lane by sending warning signals in form of sound or vibrations to the steering wheel, when deviations occur. The system can also steer the car back to the lane by applying a small force to the steering wheel. During the development of such a system, the following artifacts may be produced:

- A high-level description of the lane keeping system as a whole and which other systems it interacts with, e.g., the steering system. The results of this can be stored as textual requirements (Figure 1.1 top left) as well as a system model (Figure 1.1 top right). A system model is an abstract description of different system components and their connections.

- The system model is then broken down into discipline-specific subsystems that can be handled by the different domains, for instance mechanical, electrical, electronic and software.

From a software perspective further artifacts produced are:

- Software requirements, which can be in form of free text (Figure 1.1 mid left) or formal models such as use case models.

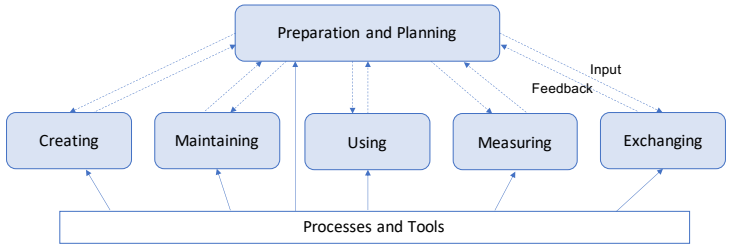


Figure 1.2: A traceability process model showing the different categories of traceability activities.

- Design models such as structural models (models that describe the software components and their connections, Figure 1.1 mid right) and behavior models (models that describe the control flow of the software) such as state charts.
- Implementation in the form of code (Figure 1.1 bottom left).
- Tests such as unit tests (Figure 1.1 bottom right) and integration tests.

Traceability in such a scenario will be the ability to relate a system requirement to its component realization in the system model, the software requirements it affects, its corresponding software components in the software model, the code that realizes this requirement and the tests validating the requirement. It may also include the ability to relate artifacts like change requests, task tickets and bug reports to the specific development artifact that concerns them, depending on if this information is later useful.

Figure 1.1 shows a simplified example of the development artifacts produced and the lines between the artifacts represent the traceability links that can be established between them.

1.2 Traceability Strategies

Successful traceability rarely happens by chance in companies but requires careful planning [10]. A plan of action for why and how traceability should be managed in a company is known as a traceability strategy. A traceability strategy consist of all activities involved in traceability planning and traceability management. The strategy therefore includes the traceability process and tools that will be used in the company. To describe the different activities involved in the traceability process, we derive a traceability process model depicted in Figure 1.2. This model is inspired by a generic traceability process model by Gotel et al. [10] which has further been used in several traceability research projects to describe traceability activities. The model contains the four activities from the model in [10] which are preparation & planning, creating, maintaining, and using traceability and two additional activities discovered during the research which are measuring and exchanging traceability information. This model has also been used as a basis for the study reported in Paper A (Chapter 3).

1.2.1 Preparation and planning

The first activity in the model is preparation and planning for traceability. This activity includes tasks such as, eliciting traceability goals of the development organization, searching for fitting solutions, acquiring tools, documenting and disseminating the traceability strategy. It is also important that the traceability strategy is assessed periodically and improved according to the needs of the organization [10]. Traceability strategies differ from organization to organization and even from project to project. This is because the strategy depends on various factors such as the specific traceability goals for the organisation/project, the development processes and workflows used, artifacts created and the tools used in the development life cycle. As a consequence, the preparation and planning phase needs to take all these factors into account as one cannot simply reuse traceability strategies from other organisations without any tailoring [17] [18]. In this thesis, we propose a methodology for designing company/project specific traceability strategies in Paper F.

1.2.2 Creation of trace links

The second activity in the model is the creation of traceability links. In many organizations, trace links are created manually [19]. However, automation is also possible through various techniques such as information retrieval [20], machine learning [21], model-based approaches [22], rule-based approaches [23] and in recent years deep learning [24]. These approaches show promise but they are still not perfect, as the techniques can miss true links and the set of candidate trace links produced contain false positives. This makes it hard to adopt for industry especially safety regulated industries [25]. To combat this problem, a human analyst is needed to check whether the generated candidate trace links are correct to come up with a final set of trace links that can be used in development. This process is known as the **trace link vetting process**, and is illustrated in Figure 1.3. In this thesis we have investigated how to improve the trace link vetting process in two studies (Paper D and E).

Trace links can be classified as being implicit or explicit. Implicit traceability refers to traceability that is established based on conventions e.g., naming conventions. For instance, a traceability link exists if the name of a component in the component model is the same as the name of a class in the corresponding implementation. While implicit traceability may seem easy to establish, it is hard to enforce conventions, as conventions can be violated leading to either no links at all or existence of a partial set of potentially erroneous traceability links [26]. Explicit traceability means that the links between the different artifacts are created and represented in some form. Explicit links can be stored as internal trace links, i.e, represented within one of the artifacts traced to, or external trace links,i.e., represented as separate artifacts [27]. The advantage of explicit links is that they are easier to assess with tools and can be provided as input to visualization tools. Where implicit links exist, automation techniques such as information retrieval techniques can be used to make the links explicit. The quality of the resulting explicit links will depend on the performance of the algorithms used, the quality of the implicit links (i.e., the conventions of creating implicit links and if they were followed) [28] and the quality of the vetting process.

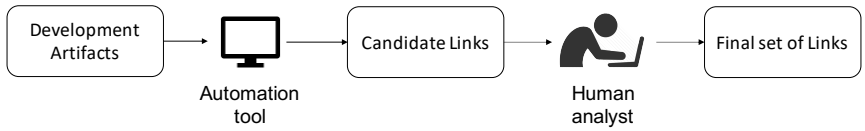


Figure 1.3: The trace link vetting process.

1.2.3 Maintenance of trace links

The third activity is maintaining the traceability links which means making sure that the links are up-to-date. As artifacts connected by traceability links evolve, the traceability links also need to be updated. Otherwise they quickly become outdated and therefore useless [29]. Maintenance of traceability links without tool support is difficult because artifacts constantly evolve. While Version Control Systems (VCS) can check for changes in the artifacts, traceability tools need to use these changes to also check if they affect the existing traceability links. Current tools that support traceability, for instance, IBM DOORS[1], provide a notification framework that allows users to be notified of changes to artifacts that are connected by traceability links. This makes it easier for the users to know which links they have to update. In this thesis, one of the contributions is a set of important factors and guidelines for traceability maintenance aimed at developers of traceability management tools and users looking to select traceability management tools (Paper B). These guidelines go beyond proposing implementation of notifications by providing guidance on how to handle versioning, tool boundaries, defining trace link semantics and consistency management.

1.2.4 Using trace links

The fourth activity in the model is using traceability. As mentioned in our definition of traceability, the links created need to be useful during the development of the system. Having the traceability links in place is not enough; tool support is needed for navigation and visualization purposes to make sure the links can be used. For instance if a product manager wants to see how many requirements already have tests, it will be very difficult to navigate through all requirements one by one to their tests. The traceability tool should be able to provide this report. The traceability strategy therefore should also elicit how these links will be used and provide the required support.

1.2.5 Measuring traceability

The fifth activity in the model is measuring of existing traceability There are two perspectives of measurements for traceability; measuring the quality of the maintained trace links, and measuring the usefulness/benefits of the existing traceability. With respect to the quality of trace links, there are two relevant metrics; completeness and correctness. The definition of completeness

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

varies and is defined differently in different development organizations. For instance, a completeness definition could be that every test has a link to a requirement. This can then be measured by checking if indeed all tests are linked to requirements. Correctness on the other hand is harder to check for, as it relies on domain-specific semantics and also human interpretation of the artifacts. Semantics can be formalized and checked by tools through definition of domain-specific traceability information models. For instance in the case of linking tests to requirements, a tool can check if the link actually connects a test and a requirement. What is difficult to check is if the requirement is actually relevant for to the test it is connected to.

Measuring traceability does not end at the quality of the links but also includes measuring the benefits of traceability. For instance, Ingram and Riddle [30] discuss the cost and benefit of traceability and insist that companies should be able to measure how much traceability management costs and weigh the costs against the benefits that traceability yields. It is difficult to properly measure the benefits of traceability since the benefits are first of all long-term, i.e., can happen years after traceability has been established and are usually not quantifiable. This should however not stop organizations from putting measures in place to evaluate how the current traceability meets the traceability goals of the organization. Even qualitative analysis of the usefulness of existing traceability can lead to improvement in the traceability strategy. Priority should be given based on value, to create and maintain trace links that yield the most benefit.

1.2.6 Exchange of traceability information

The sixth activity refers to situations where different organizations or different departments in the organization need to exchange traceability information. Due to the numerous tools used in different departments and different companies, there is a need for development organizations to plan for how traceability information will be shared and exchanged. This may mean agreeing to use certain tools that are compatible with each other. Other issues that need to be considered are technical and legal issues on whether the different organizations have access to artifacts that are traced to [31].

This thesis uses the activities described above (Section 1.2.1–1.2.6) as a systematic model to assess the state of the art and state of practice of traceability challenges and solutions as reported in Paper A. We structured our research methods to systematically elicit the challenges as well as solutions w.r.t to each of the activities. Additionally, in Paper F, where we propose a traceability introduction methodology (TracIMo), we use these activities to reason about how particular traceability strategies which define when and how these activities should take place should be defined in companies.

1.3 Traceability Management Tools

As discussed in Section 1.2, tooling is an important aspect as it enables the different traceability activities to be carried out. There are several tools that are used for supporting traceability activities and according to [16] they can be

classified into three main categories: dedicated traceability management tools, life cycle tools and general purpose tools. Based on our own research, we add a fourth category which is standalone traceability tools. This fourth category is also supported by [32] and [8]. The four categories are defined as follows:

Dedicated Requirements Management Tools – These are tools whose main purpose is requirements management. Even though the main functionality of the tool is requirements management, they may have functionality to facilitate traceability from requirements to other artifacts such as design models, task tickets and code. An example of such a tool is IBM Rational DOORS, which is a requirements management tool with capabilities to connect to other artifact providers through Open Services for Life Cycle Collaborations (OSLC) [33].

Application Life Cycle Management Tools – This category of tools provides functionality for creation and management of the different development artifacts in the development life cycle. Life cycle tools support requirements management, design, implementation, testing and other development activities. All artifacts are stored in one repository and traceability between the different artifacts is provided. The advantage of life cycle tools is that it is comparably easier to establish traceability between different artifacts. However in many cases companies have a variety of tools in place due to fear of being dependent on one vendor, unwillingness to use a certain development methodology (e.g., model driven development) or the fact that developers prefer tools that are task-focused over generic ones [16, 22, 34]. Examples of life cycle tools are Systemweaver[2], Siemens Polarion[3] and IBM ALM[4].

General Purpose Tools – These are tools that are designed to be used for many purposes. These tools can also be used for traceability. For instance, one of the most common general purpose tools used for traceability is Microsoft Excel. It is used to create, for instance, requirements to tests traceability matrices where requirements names or IDs are listed in the first column and tests in the first row. A mark is placed in the cell where two artifacts intersect if there is a trace link between them. The advantages of general purpose tools are that they are widely available and provide a cheaper traceability option for organizations. However, the disadvantage is that the tools do not scale for traceability maintenance. The tools are not aware of the actual artifacts and traceability is managed separately from the artifacts. The chances that the links and the actual artifacts become inconsistent is high.

Standalone Traceability Management Tools – These are tools that are built for the purpose of managing traceability only. Such tools need to be able to integrate and access artifacts from the development tool chain to allow creation of traceability links. They also need to provide notification mechanisms when artifacts in the different tools change in order to facilitate maintenance of the links. These tools use tool

[2] `http://www.systemweaver.se`
[3] `https://polarion.plm.automation.siemens.com`
[4] `https://www-01.ibm.com/software/rational/alm/`

integration technologies such as OSLC [33] or Eclipse Modeling Framework (EMF) [35] to integrate the different tools. The advantage of such tools is that they are configurable to include various artifacts and therefore do not force a company to change their current tools. The disadvantage is that the customization requires effort and may be costly especially in a company where tools are added or changed frequently. An example of such a tool is Yakindu Traceability[5]. In this thesis, one of the contributions is an open source standalone traceability management tool, Eclipse Capra[6].

1.4 Research Scope

Traceability is a topic that has been researched for a long time, with empirical publications in software engineering research dating back to the 1990s [14]. Our first goal was therefore to understand what the current challenges and solutions that exist in practice are, in order to know where to focus our research efforts. The first research question that this thesis answers is as follows:

RQ 1 What are the current traceability challenges and solutions in practice?

The study conducted to answer **RQ 1** (Paper A) identified several traceability challenges in different areas such as tools, processes, knowledge, measurement of traceability and so on. This thesis covers the following four challenges:

[a] High manual work involved in creating and maintaining the trace links.

[b] Lack of configurable tools.

[c] Diverse artifacts and tools used in the development life cycle.

[d] Unclear traceability processes

The challenges are investigated from two perspectives: the process perspective which investigates how different traceability activities need to be carried out; and the tool perspective which investigates how tools can be improved to support the different traceability activities.

With regards to the first challenge, we identified that the manual work involved in traceability is firstly in the creation phase where trace links need to be created and secondly in the maintenance phase where trace links need to be maintained as the artifacts they connect evolve. In the first part we investigated how tools facilitate the maintenance of trace links given that organizations already spend a lot of effort creating them. For this we strive to answer the following research question:

RQ 2 What are the primary factors that affect how and to what extent a traceability management tool can provide traceability maintenance?

The study (Paper B) not only revealed which factors are important, but based on empirical evidence, we were able to derive guidelines for traceability tool developers to develop tools that will allow for efficient maintenance of

[5] `https://www.itemis.com/en/yakindu/traceability/`

[6] `https://projects.eclipse.org/projects/modeling.capra`

traceability links. In addition, the guidelines also show which factors need to be considered to support diverse artifacts and tools, which is also one of the challenges we identified.

Based on results from the study that answers **RQ2** (Paper B), we implemented a traceability management tool with two purposes in mind: (i) to show how the guidelines we proposed can be implemented in practice, and, (ii) to be able tackle the tool related challenges [b] lack of configurable tools, and [c] diverse artifacts and tools. With this, the thesis answers the following research question:

RQ 3 How can a traceability tool be implemented in such a way that it is configurable and extendable?

The implementation and details of the tool is reported in Paper C.

Additionally, the thesis covers the challenge of high manual work involved in creating trace links (challenge [a]). While this emerged as a practical challenge that practitioners still face, the scientific community has already explored how to automatically generate trace links (e.g., [21]). Although there are some promising results in this area (e.g., [24]), the resulting candidate trace links are still not 100% correct and therefore always require a human analyst to verify the generated candidate links. The combination of automatic generation of trace links with a human in the loop is a promising way to transfer this research into practice. However, the vetting process (which is the process where a human analyst verifies automatically generated links) is flawed, as shown by Cuddebak et al. [36], since the human analyst can make mistakesby accepting false links and rejecting true links. Our aim was therefore to investigate how to improve the vetting process and the following research question was derived:

RQ 4 How can the trace link vetting process be improved?

For **RQ 4**, there are various areas and possibilities for how the vetting process could be improved. In this thesis, we concretely investigated two things; i) how providing context information to the human analyst improves the vetting process (Paper D) and ii) how gamification (defined as "usage of game design elements in non-game contexts" [37]) can improve the vetting process (Paper E). We therefore answer the following sub-research questions:

RQ 4.1 What context information is useful to human analysts when vetting trace links?

RQ 4.2 To what extent does this information help analysts to make correct decisions?

RQ 4.3 What is the impact of gamification on the vetting process?

The study in Paper A also identified challenge [d] companies have an unclear traceability strategy. A traceability strategy is a plan of action that defines the traceability activities, how they will be carried out and when and how they achieve the goals of the organization. In Paper F, we investigates how to solve this challenge by answering the following research question:

RQ 5 How can a traceability strategy be established to achieve the specific goals of an organization?

In relation to the two goals previously stated, **RQ 1** is directed towards achieving **Goal 1:** Identify current traceability challenges and solutions. **RQ 2 up to RQ 5** achieve **Goal 2:** Explore and propose solutions for the existing traceability challenges related to tools and processes.

1.5 Related Work

Due to the cross-cutting nature of the traceability topic and the breadth of this thesis, there is a vast amount of related work available. This section gives a brief overview of existing research covering the thesis topics. It also describes where the contributions of the different appended publications fit in the existing body of knowledge on traceability.

1.5.1 Traceability challenges

Several studies in literature acknowledge the challenges of traceability. Back in 1994, Gotel and Finkelstein [14] reported an empirical study with over 100 practitioners that identified challenges of traceability in industry. 18 years later, Gotel and a group of traceability researchers published an updated version of these challenges known as the Grand Challenges of Traceability with a vision on how these challenges can be tackled by researchers [19]. Further studies follow in this path (e.g., [9] and [8]). The most prominent challenges reported are the cost of traceability, lack of knowledge on traceability, lack of appropriate tools as well as social issues such as different stakeholder viewpoints [8], [19], [38]. Our first study (Paper A) adds to this body of knowledge by further investigating how the challenges as well as solutions already reported in literature manifest in practice. We provide a more up-to-date description of both the state of the art and state of practice of traceability challenges and solutions compared to existing literature.

1.5.2 Traceability tools

Traceability activities such as creation, maintenance and use, need to be supported by appropriate tools. A lot of tool-related research focuses on the creation of trace links and specifically how to automate this activity. The majority of the research is on using information retrieval techniques such as Vector Space Model (VSM) [25] and Latent Semantic Indexing (LSI) [39] to generate candidate links between two textual artifacts. Information retrieval techniques investigate the text in the artifacts and produce candidate links based on the similarity of the text in those artifacts. More sophisticated methods such as machine learning [21] and deep learning [24] have been investigated where a classifier is trained based on an existing set of links in order to identify new candidate links in the set of artifacts. Another line of research utilizes programmers interaction logs in an Integrated Development Environment (IDE) to automatically derive trace links between artifacts [40, 41].

While fully automated approaches are promising, they are not perfect both in terms of precision and recall. Precision refers to the fraction of true links identified by the automation algorithm. This measure determines how many links identified are actually true. Recall is the fraction of true links returned over the total number of true links in the artifact set. This measure determines how many of the true links present in the artifact set are identified as true links by the automation algorithm. High values of both precision and recall (close to 100%) are required for the trace links to be accurate. However, this has not yet been achieved and probably never will due to intrinsic noise in the underlying data. For these automatic approaches to be used, a human analyst needs to vet the generated candidate trace links, a process known as trace link vetting as described before. Various research exist to better understand this process from both the user perspective and tool perspective. For instance, Dekhtyar et al. [42] investigated several factors such as development experience and tracing experience that may influence the accuracy of trace links produced as a result of human decision making. Similar studies such as Cuddeback et al. [43], [36] and Kong et al. [44] also investigate the human analyst's behavior when vetting trace links. Additionally, several research tools e.g., RETRO [45], Poirot [46] and TraceME [47] have been developed to show how the vetting process can be supported in tools that generate trace links. This thesis contributes to further understanding of the vetting process by first investigating the information needs of the human analyst when vetting trace links (Paper D) as well as studying the impact of gamification as an attempt to make the vetting task more enjoyable (Paper E).

Another line of tool-related research is on how to select appropriate traceability tools. For instance Rempel et al. [48] define a framework for traceability tool comparison which enables classification and comparison of traceability management tools in order to help companies when selecting tools. Similarly, Gotel and Mäder [16] define step by step guidelines for practitioners to follow when acquiring traceability management tools. Kirova et al. [32] give guidelines for implementing effective traceability which also include which technical aspects to consider when implementing traceability. The existing research however, does not explicitly focus on how to best analyze tools for their traceability maintenance capabilities. A lot of research on tool-related traceability maintenance focuses on specific techniques for automating the maintenance process. For instance, research by Mäder et al. [29] [49] [50] investigate how to recognize different changes in UML models in order to update the trace links based on the changes in the model. A similar study by Rahimi and Cleland-Huang [51] investigates which changes performed on requirements and source code lead to changes in the traceability links, and how to propagate these changes automatically. Our study in Paper B therefore aims to fill the existing gap of lack of specific guidelines for tool developers and companies selecting tools w.r.t traceability maintenance. We provide important factors and guidelines specifically towards providing effective traceability maintenance. In Paper C, we show how these guidelines can be applied in practice by developing Eclipse Capra.

Eclipse Capra is a standalone configurable and extendable traceability management tool. Similar standalone traceability management tools aiming to support custom trace link types have also recently been developed for research

purposes e.g., Tarski [52] which uses formal first order logic to create custom traceability information models as opposed to Eclipse Capra which allows the definition of custom traceability information models using the Eclipse Modelling Framework (EMF). Another similar tool is EMFTrace [53] which also allows tracing to different artifact types through the concept of adapters similar to Eclipse Capra. However EMFTrace only supports model artifacts such as EMF, UML and URN [54] models and only supports EMF store[7] as the versioning back-end.

1.5.3 Traceability strategies

While it is important to conduct research on specific traceability activities e.g., creation and maintenance of trace links and specific techniques e.g., how to used model driven engineering to support traceability, it is equally important to conduct research on overall traceability strategies. Traceability strategies define the goals of the traceability needed by different stakeholders in the company and the way to achieve these goals which includes which links need to be created, when, how, by whom, using which tools and how they will be created, maintained, assessed and used. The book by Gotel et al. [10] defines a basic traceability process model that describes the different activities that need to be considered in a traceability strategy, acknowledging that these activities need to be tailored and planned for in specific contexts of the companies. In this thesis we extend this model with two more activities: measuring, and exchanging traceability information (cf. Section 1.2). In Paper F, we investigated how companies can develop traceability strategies tailored to their specific context and propose TracIMo, a traceability introduction methodology. TracIMo extends the work by Rempel et al. [55] which proposes a model to assess existing traceability in companies. Similar work on tailoring traceability strategies for different contexts includes the work by Dömges and Pohl [18] which proposes a framework for designing project specific traceability strategies. This framework is similar to TracIMo, but is tool-oriented, defined on an abstract level, and does not discuss how to measure and evaluate the designed strategy. Espinoza and Garbajosa [56] propose a traceability information model for defining company-specific traceability strategies. TracIMo also describes how to define custom traceability information models but goes a step further by describing how the model needs to be aligned with the process and selection of appropriate tools. Similarly, there are several case studies investigating how to implement traceability in different contexts, e.g., Durrani et al. [57], Arkley and Riddle [17], Asuncion et al. [58] and Kirova et al [32]. These studies provide experiences, lessons learned and guidelines on how to tailor traceability. However, to the best of our knowledge there is still no defined framework to systematically establish traceability.

1.6 Research Methodology

To achieve the goals of the thesis, we used a combination of different research methods. We summarize the entire research process of the thesis in two research

[7] `https://www.eclipse.org/emfstore/`

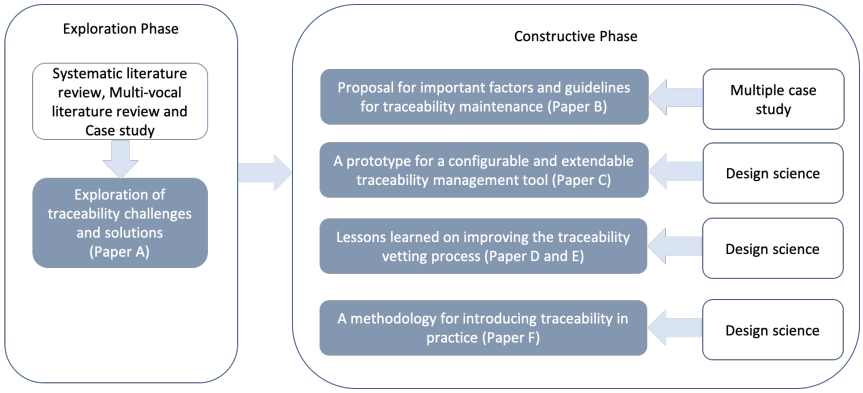


Figure 1.4: Research Methodology

phases: phase one is exploratory where the purpose was to understand the traceability topic, specifically the challenges and solutions that already exist; and phase two is the constructive phase where we construct solutions to the challenges we identified in the exploration phase. This section gives an overview of the different research methods used in the two phases. Details of how each study was carried out are given in the individual papers that make up the thesis presented in Chapter 2 to Chapter 7. Figure 1.4 shows which methods were used for which studies. This is further explained below.

1.6.1 Exploration Phase

In the first phase the aim was to understand the current challenges of traceability in order to identify areas that need further research. To achieve this, we conducted an exploratory study to systematically identify these challenges. We used a combination of three research methods: (i) a tertiary literature review, (ii) a multi-vocal literature review and (iii) a case study. Figure 1.5 gives an overview of the research process in the exploratory phase.

Systematic literature reviews: Systematic literature reviews are used to thoroughly understand what has been published on a certain topic and identify research gaps. When current literature reviews in the topic exist, they can be used as starting points for researchers. Due to the existence of recent literature reviews on traceability e.g., [59–61], we conducted a systematic tertiary literature review on existing literature reviews with the goal of identifying traceability challenges and solutions. A tertiary literature review is a systematic literature review conducted on secondary studies (i.e, existing systematic literature reviews). It follows the same protocol as a systematic literature review [62] which consists of five steps; definition of research questions, conducting the search, screening of papers, keywording using abstracts, and data extraction and mapping process. Since our aim was to specifically look for challenges and solutions, the research question is defined as follows:

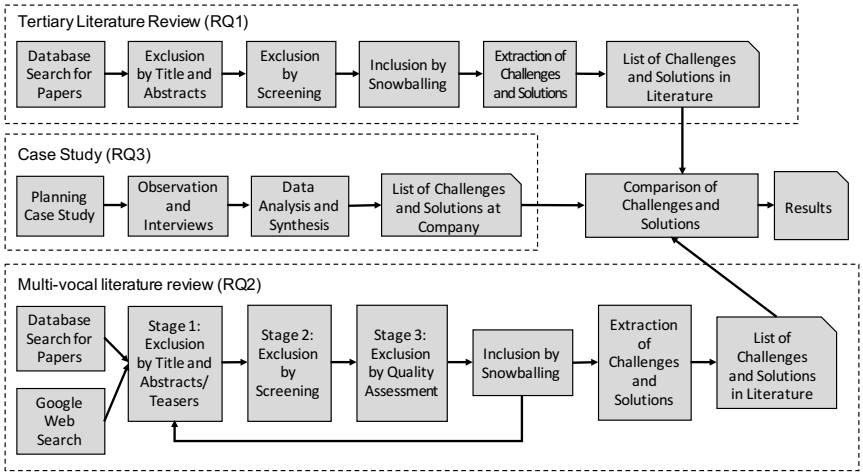


Figure 1.5: Research method for the exploration phase.

RQ A.1: What are the general traceability challenges and solutions reported in literature?

We searched for systematic literature reviews published in the topic of traceability in three scientific databases; Scopus, ACM Guide and IEEE Xplore. The papers found were screened by reading through the titles and abstracts to determine which papers were relevant. From the relevant papers we also conducted snowballing to specifically look for papers focused on traceability challenges. To extract the data from the set of relevant papers identified (24 papers), we read the papers and used a similar model to Figure 1.2 to identify challenges and solutions, and map them to the different activities in the model.

Multi-vocal literature review: While systematic literature reviews are helpful to understand the state of the art of a certain topic, it is difficult to understand the state of practice from such studies. To address this challenge we performed another form of literature review called multi-vocal literature review. A multi-vocal literature review (MVLR) [63], is a systematic literature review that includes both scientific literature as well as grey literature. Grey literature includes literature not published in scientific venues and therefore not peer reviewed such as white papers, blog articles, industrial conference presentations, etc. While MVLR is not such a common literature review method in software engineering, it is recommended when researchers are studying a practical topic and therefore looking to understand both the state of art and state of practice [63]. We conducted a MVLR to identify traceability challenges reported from the perspective of the automotive domain. This means that we limited our search to scientific work and grey literature published in the context of the automotive domain. We used similar steps as the ones used in the tertiary literate review described before. Two research quesions were formulated:

RQ A.2.a What are the challenges and solutions regarding traceability when addressing the demands of automotive standards ASPICE and ISO 26262?

RQ A.2.b What are additional relevant traceability challenges and practices in the automotive industry?

For each research question a separate search was conducted to identify relevant sources. We searched Scopus, ACM Guide and IEEE Xplore to identify scientific literature and used Google Web search to find grey literature. For scientific papers, screening was done by reading the titles and abstracts, and for the grey literature, we screened the sources in 3 stages. In Stage 1, we read the title and short description given by the Google web search and excluded papers which did not follow the criteria below:

- The source is not an encyclopedic article
- The source is publicly available
- The source is written in English or in German
- The source was not written by the three authors of Paper A.

In Stage 2, we used available information about the source, e.g, its full text and its meta-data to further exclude sources based on the criteria above, if this information was not discovered in stage 1. In Stage 3, we evaluated the quality of the source and excluded sources that did not have any useful information on challenges or solutions of traceability in the automotive domain. For example, all sources that only advertised traceability tools but did not provide any information on the traceability concepts or contexts of the tool were excluded. This process led to a total of 245 relevant papers, 120 from the search for RQ 2.a and 125 from the search for RQ 2.B. The sources were analyzed using the model in Figure 1.2 to map challenges and solutions to the different activities. This MVLR enabled us to gather another set of traceability challenges and solutions reported by academia as well as industry practitioners such as traceability tool vendors and traceability tool users.

Having the challenges from the two literature sources, we investigated how these challenges manifest in practice. To achieve this, we conducted a case study with a large automotive supplier developing embedded systems.

Case study: A case study is an empirical research method that is aimed at investigating a contemporary phenomenon in its context [64]. A case study can be descriptive, exploratory, explanatory or improving. We conducted an exploratory case study. An exploratory case study is aimed to collect information on what is happening so as to generate new research questions [64]. The aim of the case study was to provide empirical evidence of the challenges and solutions found in the literature, show how these challenges manifest in practice and identify new challenges that were not reported in literature. The study was conducted with a large automotive supplier that follows A-SPICE [65] and is therefore required to implement traceability. We selected the particular company

because it is in a domain where traceability is mandated and therefore was already using traceability and looking into how to improve the current traceability practices. Selecting a company which is not required to have traceability may not have revealed the real challenges as the rigor in which traceability is established and used may be low. The case study answered the following research question:

RQ A.3: Which of the traceability challenges reported in scientific literature and non-scientific literature are also evident in practice in the automotive domain and how have they been solved?

The study involved seven participants of three different roles (two senior traceability experts, four system architects and one developer), from two embedded systems development departments in the company. The data collection was in form of semi-structured interviews where the collected data was later analyzed systematically using thematic coding to identify traceability challenges and solutions. The details of the tertiary literature review, the multi-vocal literature review and the case study are reported in Paper A (Chapter 2).

1.6.2 Constructive Phase

Constructive research involves the creation of solutions or constructs to solve a practical problem and reasoning on how the solution contributes back to scientific theory [66]. The constructs created as part of constructive research are, e.g., models, guidelines, processes, designs and so on. It is a common approach used in social sciences but also used quite often in software engineering research [67]. The second phase of the thesis research process is described under the constructive research umbrella since in this phase, the aim was to develop solutions for the challenges identified in the exploration phase. For this we applied a number of research methods:

Multipe case study: As previously mentioned, a case study investigates a phenomenon in its context. In our second study (Paper B), we conducted multiple case studies to understand traceability requirements from a tooling perspective and further understand the challenges of traceability in general. We conducted interviews with nine of our industrial and academic project partners to elicit requirements for a traceability tool, and 24 software development stakeholders from 15 industrial cases to provide a broader overview of the current state of the practice on traceability management. The data was analyzed to specifically look for concepts related to traceability maintenance. From the analysis we formulated important factors and guidelines for traceability maintenance for traceability tool developers and for companies who want to acquire traceability tools. We evaluated our guidelines through further interviews with three experts on traceability tools. The guidelines together with the details of how the study was conducted are reported in Paper B (Chapter 3).

Design science: Design science involves design and investigation of artifacts in context. It is an iterative process in which the researchers conduct several cycles of problem investigation, implementation of artifacts (e.g.,

software prototypes) that will solve the problem and evaluation of the artifacts in a given context to find out if the artifacts solve the problem [68]. In the thesis, design science is used in four studies, Paper C, D, E and F. Table 1.2 summarizes the different methods used in the design science activities, i.e., problem understanding, implementation and evaluation of each study. In paper C, the aim was to implement a traceability management tool based on requirements from our industrial partners and to show the feasibility of the guidelines we propose in Paper C. To understand which requirements are relevant, we conducted formal and informal interviews with the industrial partners. In the implementation phase we implemented Eclipse Capra [69], a traceability management tool based on the gathered requirements. The implementation was done in several phases and different versions of Eclipse Capra were demonstrated to project partners during meetings for feedback. The final artifact of this study is a traceability management tool that is now available and maintained as an open source traceability tool.

In paper D, the study investigated how providing context information to the human analyst improves the traceability vetting process. This study took a design science approach where, to understand the problem in detail, we conducted interviews with ten traceability practitioners. In the interviews we asked about traceability practices, explained the traceability vetting process and asked for which context information is relevant for the vetting process. We complimented this with a literature review on how existing tools that support context information work. We investigated six relevant existing tools. From the interviews and the tool investigation, we collected requirements for context information needed when vetting trace links and implemented the presentation of the tool in Eclipse Capra. To evaluate our findings we conducted a controlled experiment to understand which context information is useful and to what extent. A controlled experiment is "an investigation of a testable hypothesis where one or more independent variables are manipulated to measure their effect on one or more dependent variables" [70]. Controlled experiments are useful when studying the impact of a certain phenomenon in isolation. The experiment had 33 participants which were divided into two groups, one with 16 and one with 17 participants. The task was for each participant to vet a set of automatically generated candidate trace links. One group used a version of Eclipse Capra which did not include the display of context information such as the artifact metadata, while the other group used a version of Eclipse Capra which contained this information. We used statistical analysis to analyze the performance of both groups in terms of the speed of vetting links and the accuracy (precision and recall). We also used a pre-experiment questionnaire to collect data on the participants background and experience and a post-experiment questionnaire to collect data on the vetting task and the usefulness of the information provided. Further details on how this study was conducted are given in Paper D (Chapter 5).

In Paper E, we wanted to understand particularly the impact of different gamification features on the analyst during the vetting process. To

understand the problem, before the experiment, we conducted a survey where we asked several participants who participated in the experiment reported in Paper D to rate which gamification features will be useful for the vetting task. From the results of the survey, we selected levels and badges as useful gamification features. In the implementation phase we extended Eclipse Capra to include levels and badges during the vetting process. To evaluate the impact of gamification, we set up a controlled experiment to investigate the effect of gamification on four dependent variables; speed, accuracy and perceived enjoyment of the task as well as the usability of the vetting tool. We recruited 24 students to take part in the experiment where the students were randomly divided into two groups of 12 each. One group performed the vetting task using Eclipse Capra with no gamification while the other group used the gamified Eclipse Capra. The experiment task was to vet automatically generated trace links in a 45 minutes time window. If the participants think a link is true then they mark it as accepted, if they think it is false they mark it as rejected. We used statistical analysis on the results of the experiment to determine the impact of gamification on speed, accuracy, perceived enjoyment of the task, and usability. To capture perceived enjoyment, a post-experiment questionnaire was used to ask participants on how they experienced the vetting task. We also collected data on the usability of both versions of Eclipse Capra using the system usability scale (SUS) questionnaire [71].

The fourth study which used design science is paper F. In this study, the aim was to propose a methodology for introducing traceability in industry. We worked together with a company where we conducted iterations of understanding their problem, implementing a solution and evaluating the solution at the company. We used semi-structured interviews to understand the problem at the company and conducted a literature search to look for already existing solutions. Instead of creating a traceability introduction methodology from scratch, we extended Rempel et al.'s [48] framework for assessing existing traceability in industry which already contains steps that can be reused in the creation of traceability strategies. In collaboration with the company, we extended the framework to include steps that make it possible to plan for and deploy a traceability strategy in industry. The resulting artifact is TracIMO, a traceability introduction methodology. To evaluate TracIMo, we conducted a case study at the company to design and deploy a traceability strategy. We used the case study to evaluate both the application of TracIMo as well as the advantages of the designed and deployed traceability strategy in the company. Further details on how this study was conducted are reported in Paper F (Chapter 7).

1.7 Threats to Validity

In this section, we describe the threats to validity for the entire thesis, taking into account the different research methods used in the different papers. We use the categorization of validity threats given by Yin [72] and later by Runeson et

	Design science Activities		
Paper	**Problem understanding**	**Implementation**	**Evaluation**
C	Interviews Focus groups	Tool implementation	Demonstrations
D	Interviews Investigation of existing tools in literature	Tool implementation	Experiment
E	Survey	Tool implementation	Experiment
F	Interviews Investigation of existing methodologies in literature	Development of TracIMo	Case study

Table 1.2: Different research methods used in the three design science studies (Paper C, D, & F)

al. [64] which are specific for case studies but also applicable to reason about validity threats of other research methods. A more detailed description of the threats to validity can be found in the individual papers reporting the studies.

1.7.1 External validity

External validity is about how much we can generalize the findings of a study. It calls for the researcher to reason about the specific context in which the study was conducted in order to contemplate about the applicability of the results in other scenarios. In Paper A, we used a tertiary literature review, a multi-vocal literature review and a case study to identify existing traceability challenges and solutions. We applied three different methods to achieve both method and data triangulation in order to minimize validity threats. However, w.r.t the tertiary literature review, the last systematic study we included was published in 2014 with the latest primary study published in 2013, so there is a chance that we missed newer challenges reported after 2014, however, the snowballing as well as the multi-vocal literature review covers literature of up to 2017 thus minimizing the threat. With regards to the case study, we interviewed different roles to get a holistic view of the challenges and solutions in the company, however since we only used one company, we cannot generalize that the challenges and solutions identified are also experienced in the same way in other companies.

The constructive phase of our research aims at developing solutions that can be widely applied. However our aim was also to propose solutions and investigate their feasibility given specific contexts. The general strategy we used to maximize external validity for the five studies is data triangulation. We collected data using different methods, from different roles and different companies where possible. In Paper B, we only used interviews but had three sets of interviews, one specifically eliciting traceability requirements and one focused on general traceability practices and a final one to validate the findings.

All the interviews involved participants with different roles, from different companies and from different domains with the exception of the first set of interviews where all participants where from the automotive domain. Similarly for Paper C, we collected requirements from various project partners from different companies with different roles and our validation was done using tool demonstrations. In Paper D, we conducted interviews to identify information needs for the human analysts by interviewing ten traceability practitioners of different roles, from different companies and different countries. We also complemented the data collected from interviews with a literature search of existing traceability vetting tools. Additionally, the results were validated using a controlled experiment with a mix of students and industrial subjects.

The generalizability of paper E and F is particularly low. In paper E we used a controlled experiment with 24 students. While the use of students in experiments has been criticized in software engineering research, it is still a valid method for simplification of real life scenarios in order to learn more about a specific variable [73]. However, the number of subjects used in the study is also low which limits the generalizability of the study. In paper F, where we propose a traceability introduction methodology, the study to derive the methodology as well as the validation was conducted in one company and therefore further cases studies are needed before we can generalize our findings.

1.7.2 Internal validity

Internal validity is usually relevant when a causal effect is under study [64]. Internal validity poses the question of whether there are other external factors affecting the causal effect under investigation that the researcher is not aware of. In controlled experiments, it is important to consider internal validity in order to draw meaningful conclusions. In both Paper D, and Paper E, we used controlled experiments. For both studies, we collected data on the confounding factors that could have an influence on the results. Such confounding factors are e.g., the tracing experience, programming experience and knowledge of the system used. We collected data on these factors in a pre-experiment survey and took the data into account during analysis. For paper E, we opted for higher internal validity and used only students in the experiments. Confounding factors were also recorded and used in the analysis.

For the studies where we used interviews (Paper A, B, C, D, and F), our conclusions are drawn from the results of the interview. A threat to internal validity exits if the there is a chance that the interviewees did not give honest answers with the fear that the answers would reflect badly on their companies if the results were published. To mitigate this, for all the studies, we guaranteed the anonymity of both the interviewees and the company when publishing the results.

1.7.3 Construct validity

Construct validity reflects to what extent the phenomena being studied are understood by both the researcher and the subjects included in the research. In this thesis, construct validity is a potential threat to most of our studies that used interviews. When conducting interview studies, it is essential to make sure

that both the interviewer and interviewee have a common understanding of the concepts investigated and the terms used in the interviews. When there is a misunderstanding in the constructs, the results are unreliable. In all the studies in which we used interviews, we minimized this threat through the following: first we conducted pilot interviews to see if the questions in the interview guide make sense and are understandable; second, interview participants were properly selected to make sure that they are in a role where traceability is used and/or understood; third, the purpose of the study was discussed and important terms explained before the interview; and fourth, we performed member checking [74] where we sent the results of the interview back to the interview participants for confirmation and clarification.

Construct validity can also apply to controlled experiments. In our controlled experiments, before the experiment started, we introduced the traceability topic briefly to the participants, explained important terms, and gave clear instructions for what needs to be done. We also recorded the screens so that we could exclude results from participants who did not follow instructions and clearly misunderstood the experiment.

1.7.4 Reliability

Reliability questions to what extent the results of the study are dependent on the specific researchers that conducted the study. This threat is particularly relevant for interview studies where the results are dependent on the interview questions asked (interview guide) and derived through a thematic analysis. To first ensure reliability, all the interviews were piloted to check if the questions are understandable and make sense. The pilots in this case help to reduce ambiguity in the wording of the questions as well as reduce researchers' bias. To facilitate some form of replication, we published the interview guides that were used in the studies when the papers were submitted for peer review. This makes them accessible to the reviewers as well as other researchers interested in replicating the studies. With respect to the reliability of data analysis, it is hard to ensure that given the same interview transcript, two different researchers will come up with exactly the same codes. To minimize the reliability threat, we made sure that at least the initial phase of the analysis was done by multiple researchers in order to reduce individual biases. We also sent the result back to the participants (member checking), to ensure that they agree with our derived results.

To ensure a certain level of reliability in the experiments, for Paper E, we packaged the experiment in a virtual machine that is publicly accessible. We also published our experiment materials such as experiment instructions, pre- and post-experiment surveys.

1.8 Contributions

This section describes the contributions of the thesis. As the thesis is made up of six distinct publications, the section discusses the contribution of each paper in a separate subsection while a synthesis of how the papers contribute to the overall goal and research questions of the thesis is given in Section 1.4.

1.8.1 Paper A: Software traceability in the automotive domain: Challenges and solutions

To build a foundation for our research, it was important to understand the current and existing challenges and solutions of traceability. To achieve this, we conducted an exploratory study using three different methods: i) a tertiary literature review, where we examined 24 secondary studies and extracted a list of challenges and solutions they report, ii) a multi-vocal literature review where we reviewed 245 scientific and non-scientific sources, and iii) a case study with a large automotive supplier where we used interviews and observations to identify traceability challenges and solutions. The decision to conduct a case study in the automotive domain was driven by the fact that we needed to investigate the challenges and solutions in a company that already implements traceability with some rigor and this is true for automotive companies, since traceability is mandated by safety standards. Additionally, there were no studies investigating traceability challenges and solutions particularly in the automotive domain and our study fills this gap.

This study answered the following research questions:

RQ A.1: What are the general traceability challenges and solutions reported in literature?

RQ A.2: What are the particular traceability challenges and solutions in the automotive domain?

RQ A.3: Which of the reported traceability challenges in scientific literature and non-scientific literature can be observed in practice in the automotive domain and how have they been solved?

We identified challenges and solutions in four different phases of traceability inspired by the model in Figure 1.2. These phases are preparation and planning, creation and maintenance, outcome of traceability, and exchange of traceability. The challenges and solutions are summarized in Table 1.3

Challenge	Challenge Solved?	Solutions
Preparation and Planning		
Knowledge of Traceability		
Lack of knowledge about and understanding of traceability	Yes	Training, Updated guidelines from certification bodies
Difficult to define information model	Partially	Defined traceability information model, Updated guidelines from certification bodies
Level of granularity	Yes	Defined traceability information model
Unclear traceability process	Partially	Defined traceability process, Defined traceability information model, Structured information, Integrated tool platform, Tool integration

Creation and Maintenance		
Tools		
Lack of Configurable Tools	Yes	Flexible tools
Confidence in Tools	N/A	Certified Tool Suite
Inaccessibility of Artifacts	Partially	Centralized data storage, De-centralized data storage, Flexible tools
Diverse Artifacts and Tools	Partially	Integrated tool platform, Tool integration, Integrated modelling language, Structured information
Manual work	No	Automation, Just enough traceability, Integrated tool platform, Integrated modelling language
Human Factors		
Misuse of Traceability data	N/A	Training
Perceived as an overhead	No	Automation, Report generation tools, Just enough traceability
Organization and Process		
Distributed software development	Yes	Centralized data storage, De-centralized data storage
Traceability Across Lifecycle Phases	N/A	Integrated tool platform, Defined traceability process, Automation, Integrated modeling language
Reuse of Traceability Information	N/A	–
Outcome of traceability		
Uses of Traceability		
Trace links are almost never consulted or used	Partially	Report generation tools, Just enough traceability
Lack of proper visualization tools	No	Report generation tools
Measurement of Traceability		
Assessing the traceability maintained	No	Automation, Defined traceability process, Defined traceability metamodel, Structured data
Return on Investment (ROI).	No	Cost-benefit models, Just enough traceability, Automation
Exchange of traceability		
Exchange of Traceability Information		

Lack of Coordination in traceability activities	N/A	Collaboration tools, Defined traceability process
Lack of interchange standards	No	Common standard
Conflicting objectives	N/A	Defined traceability process
Confidentiality Constraints	Partially	–

Table 1.3: Challenges and solutions for traceability in the automotive domain

In total we identified a set of 22 challenges and 16 unique solutions from the perspectives of academia, tool vendors and consultants reported in both scientific and grey literature. Out of the 22 challenges, 17 were found in the automotive company, of which six were still unsolved, six had partial solutions and five were solved. Additionally, the study revealed that the challenges in the automotive domain have a lot of overlap with general traceability challenges. However, the domain has more restrictions such as traceability being mandatory due to safety standards such as ISO 26262 [75], making existing solutions in literature harder to apply in practice.

1.8.2 Paper B: Traceability Maintenance: Factors and Guidelines

From our previous study (Paper A), we identified that one of the persisting challenges is that establishing traceability is expensive as it requires manual work for both the creation and maintenance of the links. While a lot of effort is invested in companies to create the links, if the links are not updated as the artifacts they connect evolve, the links become outdated and thus useless. It is therefore important to put as much effort in maintaining the links as in creating them. Creation of the links is harder to automate or semi-automate because the initial set of links does not exist and needs to be created from scratch. Maintenance on the other hand is comparably easier to automate or semi-automate because the links and information about the artifacts they connect already exist. However, the extent to which tool support for traceability maintenance can be provided varies depending on a number of factors. It is therefore important for the development organizations (when selecting traceability tools) and tool vendors (when developing traceability tools) to understand these factors.

We conducted a study to identify the important factors that affect the extent to which a traceability tool can provide maintenance support. We analyzed data from two sources: 9 interviews that were aimed at eliciting traceability requirements for a traceability solution and 24 semi-structured interviews from a study aimed at investigating traceability management practices. We formulated guidelines from each of these factors and validated our guidelines with interviews of expert users and developers of traceability tools used in industry. The study answers the following research question:

RQ B: What are the primary factors that affect how and to what extent a TM solution can provide traceability maintenance?

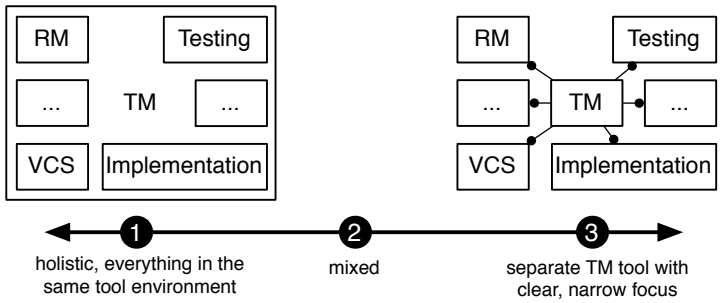


Figure 1.6: The spectrum of tool boundaries from holistic tools to complete separate traceability management tools.

Our results identified the following four influential factors of traceability maintenance:

F1: Versioning – traceability maintenance is affected by whether or not a Version Control System (VCS) is used. The granularity of the changes that can be extracted from a VCS influences the extent to which traceability maintenance can be supported. From this factor, we suggest the following guidelines: G1) Version your traceability model just like all other artifacts; and G2) Ensure that you are able to extract explicit deltas for all models from your chosen VCS.

F2: Tool Boundaries – In a practical scenario, it is expected that multiple tools are used to manage the different development artifacts. Maintaining traceability between artifacts in different tools is difficult as the tools are not connected to each other in any way. We elicited three tool-boundary scenarios that are possible (cf. Figure 1.6). One scenario is where a holistic tool is used to support manipulation of all artifacts. While traceability becomes easier in this case, in practice finding a holistic tool that supports all development activities is difficult. The second scenario is having a dedicated traceability management tool to connect the different tools. While this is feasible, it requires that the tool is able to connect to the existing tools in the development tool chain. This can be done through tool adapters. However, for each new tool, an adapter needs to be created which is potentially not a trivial task. A third scenario is a mixed scenario where a traceability tool is combined with tools for other functions e.g., requirements management tools. This scenario inherits all the negative aspects from the other two scenarios. From this factor we derive the following guidelines: G3) Traceability tools should provide well defined interfaces and easy, direct access to managed traceability links; G4) Aim for either a holistic solution or a completely separate traceability tool with a carefully designed tool adapter concept, avoid combining strategies; and G5) Use a common standard as "glue" to simplify the development of adapters.

F3: Configurable Semantics – The degree to which a traceability metamodel/information model captures domain-specific semantics and whether the

links are explicit or implicit influences how the consistency of traceability can be defined. This is because consistency must be defined and tailored to specific domains. For instance, to ensure that a traceability link from a requirement to a test is truly linking a requirement and a test, one needs to be able to define the semantics of the concept of a requirement and a test within the link in the traceability metamodel. From this factor we derive the following guidelines: G6) Avoid implicit, convention-based traceability links and strive instead for explicit links that can be checked with tool support; and G7) Prefer a domain-specific, semantically rich traceability metamodel as this simplifies traceability maintenance.

F4: Consistency Specification – The solution space for consistency specification spans two dimensions: the process by which traceability is maintained and the type of functions that are used to maintain traceability. From the process dimension, there are two ends of the spectrum: 1) traceability is maintained in a top-down approach and inconsistencies in traceability links are fixed immediately and 2) a bottom-up and ad-hoc process of maintaining the traceability links where the links are fixed on demand is used. From this dimension, we derive the following guideline: G8) Ensure that your TM solution supports a flexible combination of both top-down and bottom-up maintenance approaches.

The other dimension of consistency specification is the type of consistency function specified. This can be manual, semi-automatic or automatic. For this dimension, we derive the following guidelines: G9) Support an integrated mix of manual and complementary automated approaches to consistency specification; and G10) For automatically generated links, prefer no links at all over (possibly) inconsistent links.

1.8.3 Paper C: Capra: A Configurable and Extendable Traceability Management Tool

From the previous studies, we gathered knowledge on the challenges of traceability and specific requirements for what a traceability tool should contain. Analyzing the requirements, we discovered that there is no solution that could cover all the requirements as the requirements greatly vary depending on the development organization and are sometimes contradictory. For instance, different companies have different requirements for which trace link types should be included in their traceability information model based on the artifacts in their development environment. Additionally, technical requirements such as where the links should be stored (e.g., in a database, as a model or as a simple text file) also vary. With these requirements in mind we developed a traceability solution that can be customized and extended to support the diversity in the requirements. The study used design science where we gathered requirements from both our industrial and academic partners, implemented a prototype and evaluated the prototype through demonstrations. From these demonstrations we got feedback and more requirements for another iteration. The resulting tool Eclipse Capra, is an open source project[8] and therefore freely available and welcomes contributions from any developer.

[8] `https://eclipse.org/capra`

Factor	Guidelines	Implementation in Eclipse Capra
Versioning	G1 and G2	Stores explicit trace links as an EMF model. The location of the trace model is configurable and the resulting trace model can be versioned using various VCS.
Tool boundaries	G3	Trace links are stored in an EMF model and accessible via well defined Eclipse extension points.
	G4	Eclipse Capra is a standalone traceability management tool.
	G5	Uses EMF, a well defined and widely used modelling framework.
Configurable semantics	G6	Trace links are stored explicitly.
	G7	The traceability information model is fully configurable and allows definition of domain specific link types.
Consistency specification	G8	Supports both top-down and bottom-up approaches of creating trace links.
	G9	Supports manual creation of traceability links as well as semi-automatic maintenance of trace links by sending warnings for suspect links.
	G10	Supports automatic visualisations of internal links which already exist in specific DSLs.

Table 1.4: Proposed factors and guidelines and their implementation in Eclipse Capra.

In our solution, we followed the factors and guidelines defined in Paper B to ensure that the tool will support maintenance as much as possible but also as another way to validate some of the guidelines we proposed. As a result, the tool is extendable in three different perspectives:

The traceability link types – It is possible to create link types with different semantics depending on the needs of the development organization. The tool allows for the links to be defined in a traceability metamodel and this can be extended with domain-specific links, thus following guideline G7.

The artifacts that can be traced – Since different development organizations use different tools one cannot develop a traceability tool that is complete. There are always going to be more tools that are not supported. To

overcome this problem, our tool is a dedicated traceability management tool and therefore follows G4. It allows adding tool adapters to support different tools on a need to need basis. Therefore it is extendable. The tool uses EMF [35] as a common standard for the adapters which follows guideline G5. As adding adapters requires some technical expertise, the tool ships with adapters to support common modeling languages such as EMF and UML, programming languages such as Java, C/C++ and PHP and general purpose tools such as MS Office. Adapters for more specific formats such as ReqIF and task tickets from common task management tools such as JIRA and Bugzilla are also available. Since the tool is open source, more adapters can be contributed by the open source community. As the adapters can be turned on and off, the tool is both configurable and scalable.

The storage of the artifacts – The needs for storage of traceability links can differ depending on the development organization. For instance, in some cases the participants in our study preferred to store the links per project, while other participants preferred to store the links per workspace and others per repository. Therefore, the tool allows this to be configured. This makes it possible for the tool to adhere to G1 and G2 depending on the storage and VCS choice of the end users.

Following guideline G3, the tool also provides interfaces to expose the traceability model to various tools that, e.g., provide visualization. To show the feasibility of this, Eclipse Capra currently has two different visualization options, one utilizing PlantUML [76] and one using the Eclipse Graphical Editing Framework (GEF) [77]. Table 1.4 gives a summary of the guidelines proposed in Paper B and how the implementation of Eclipse Capra follows them.

1.8.4 Paper D: Vetting automatically generated traceability links: What Information is Useful to Human Analysts?

The study in Paper A identified manual work as a challenge. This relates to the manual work involved in creating and maintaining trace links. To solve the challenge of manual work involved in creating trace links, various automated approaches using information retrieval, machine learning and deep learning have been proposed over the last decade. However, since these approaches to not generate accurate links, a human in the loop is needed to verify the generated candidate trace links. This task is known as the trace link vetting process. Past research has shown an indication of that the vetting process can be flawed and that a human analyst can make mistakes [36]. While there is some research to better understand the vetting process, e.g., [42, 78], none of the research has investigated the information needs of the human analyst during this process. To this end, we conducted a study to fill this gap. The study answers the following research questions:

RQ D.1 What context information is useful to human analysts when vetting trace links?

RQ D.2 To what extent does this information help analysts to make correct decisions?

The study used design science, where we conducted interviews with ten practitioners who have experience with traceability to understand their information needs. The interviews reveal that human analysts need information from three different sources: 1) from the artifacts connected by the link, 2) from the traceability information model, and 3) from the tracing algorithm. We investigated existing trace link vetting tools that have been published to identify the information presented to the human analyst. We extended Eclipse Capra to support the vetting process and represent the information we collected from interviews to the human analyst. The study evaluated to what extent context information is useful using an experiment. In our study, we use the definition of context information from Abowd et al. [79]: context information refers to any information that can be used to characterize the development artifact – e.g., the meta data of an artifact, such as the date it was created or modified. In contrast, the content of the artifacts, such as the code in a Java file or the textual description of a requirement, is not considered context information. Our assumption is that offering context information together with the content of the artifacts will improve the decisions made by human analysts. We conducted the experiment with 33 participants divided into two groups of 16 and 17 participants. One group performed the vetting task without the context information elicited from the interviews and one group had the extra context information. The results of the experiment show that there was no significant difference in the performance of the two groups in terms of time spent or precision and recall. However, the experience of the analyst with the particular system matters. From this study we draw the conclusion that when selecting who should vet trace links, the person with the most experience with the artifacts involved should be selected. For example, candidate trace links between requirements and code should be vetted by a developer of the system, while candidate trace links between requirements and architecture should be vetted by an architect of the system.

1.8.5 Paper E: Impact of Gamification on Trace Link Vetting: A Controlled Experiment

In our previous study (Paper D), we discovered that since the vetting task is tedious, the participants also found it boring. Previous studies have shown that gamification can keep users performing a certain task motivated and engaged [80], [81]. We therefore hypothesize that gamification can improve the vetting task by motivating the analyst during the vetting process. We conducted a study to answer the following research question:

RQ E What is the impact of gamification on the task of vetting automatically generated trace links?

For this study, we concretely investigated the impact of two gamification features; levels and badges. We conducted an experiment with 24 participants where 12 performed the vetting task with gamification features enabled, and 12 without the gamification features. In the experiment we measured the speed

(links vetted/time), correctness, perceived usability of the tool, and perceived enjoyment of the task. The experiment showed that there is no significant difference between the speed, correctness and perceived usability of the tool between the two groups. However, the gamification features significantly increased the users' perceived enjoyment. This study indicates that on the one hand gamification does not hamper performance in the vetting task. This is important especially for correctness as we do not want users to concentrate more on the gamification features and loose focus on the actual vetting task. On the other hand, the study shows that the gamification features increase the users perceived enjoyment and therefore make the task less boring.

1.8.6 Paper F: TracIMo: A Traceability Introduction Methodology and its Evaluation in an Agile Development Team

Our first study (Paper A), shows that from the process perspective, one of the challenges is companies having an unclear traceability process. For traceability to be successful, a company needs a well defined traceability strategy. However, in practice, many factors play a role in the definition of the traceability strategy and it is not always clear which steps a company should follow in order to define their tailored traceability strategy [82], [83]. To solve this challenge, we conducted a study to answer the following research questions:

RQ F.1: How can traceability be established to achieve the goals of the organisation and yield a measurable impact?

RQ F.2: What are the short-term benefits of introducing traceability?

RQ F.3: What are the challenges and key decisions associated with introducing traceability?

The aim of the study was to create a methodology that consists of concrete steps that companies can follow to design and introduce a traceability strategy. We therefore used design science where we worked closely with a company in the finance domain who wanted to design and introduce a traceability strategy in the company. We collaborated with the company to understand the problem and conducted a literature search for existing guidelines on how to introduce traceability. Since there are no explicit guidelines on how to introduce traceability, we extended the work by Rempel et al. [55] – which proposes a model for assessing existing traceability within companies – by adding steps necessary to design and deploy a traceability strategy. We named the resulting methodology TracIMo, which is short for Traceability Introduction Methodology (Figure 1.7). To evaluate TracIMo, we conducted a case study with a company in the finance domain, introduced traceability and evaluated both how TracIMo was appropriate for designing the traceability strategy and the short term benefits of the resulting traceability strategy.

The study shows that TracIMo can be used to design and deploy a traceability strategy in companies. One major strength of TracIMo is that it applies the GQM (Goal/Question/Metrics) goal definition strategy [84] which makes it possible to derive measurable traceability goals. Table 1.5 shows an example

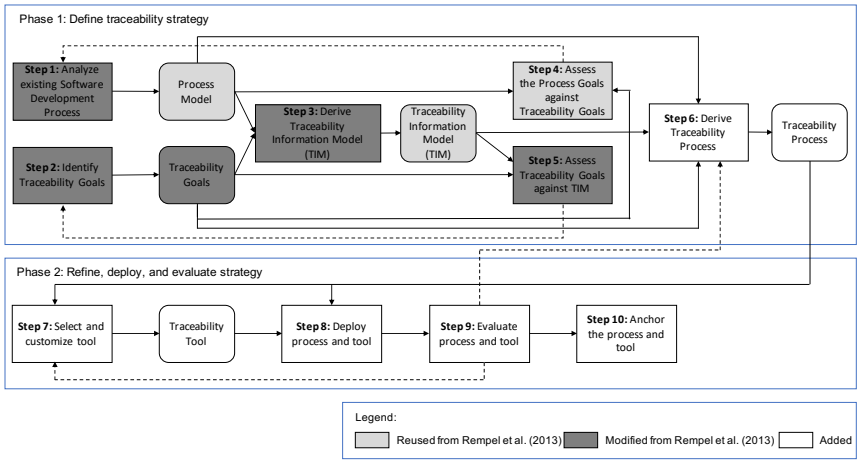


Figure 1.7: Schematic overview of TracIMo, indication which steps have been reused and modified from Rempel et al. [55].

of a measurable traceability goal defined in our case study defined from the lead developer's point of view as well as potential metrics to measure how the goal has been reached. Additionally, TracIMo is iterative and therefore allows for evaluation and re-evaluation of the different steps.

Applying TracIMo to the company, we were able to define a tailored traceability strategy as well as evaluate its short term benefits. The short term benefits identified were:

- improvement in effort estimation during sprint planning as the developers could use the existing traceability to identify how complex a task will be;
- improvement in task understandability as developers used trace links to further understand which artifacts are connnected to the task they are assigned to; and
- improvement in identification of missing artifacts, as it was now possible to notice missing artifacts when creating trace links (for example, having trace links between implementation classes and tests, made possible to know which tests are missing).

1.9 Summary of the Contributions

This section gives a summary of the contributions of the papers with respect to the overall thesis goal. We also highlight the main takeaways of the thesis based on the different studies. To recap, the main goal of the thesis is to improve traceability tools and processes which is further divided into two goals:

Goal 1: Identify current traceability challenges and solutions.

Goal 2: Explore and propose solutions for existing traceability challenges related to tools and processes.

Goal 3:	Improve the accuracy of effort estimations for tasks from the Lead Developer's point of view.
Question 1:	How much does the estimated effort differ from the actual effort?
Metrics:	• Average number of tasks per sprint • Average number of deviating tasks per sprint • Percentage of deviating tasks per sprint (derived) • Initial estimation for each task in story points • Updated estimation for each task in story points • Average increase/decrease in effort per task (derived) • Number of JIRA comments about effort per task
Question 2:	How confident is the lead developer in the estimation of tasks?
Metrics:	• Likert scale confidence 1 – not confident at all 5 – very confident per task (Questionnaire with lead developer) • Number of low confidence tasks that required a change • Number of high confidence tasks that required a change

Table 1.5: Goal/Question/Metric example for a traceability goal on improvement of effort estimation.

Paper A contributes to Goal 1 and answers **RQ 1: What are the current traceability challenges and solutions in practice?**. Through literature reviews and a case study, we identified 22 traceability challenges and 16 unique solutions. The study showed that there is a great overlap between general traceability challenges and those experienced in the automotive domain even though conditions such as safety certification are more prominent in this domain than other domains, e.g., the finance domain. Examining the applicability of the solutions in practice, we identified six challenges that are still not solved and call for further research on new solutions and enhancement of existing solutions. The main takeaway from this study is that *there are still a lot of traceability challenges where proposed solutions do not work due to practical challenges such as scale, existing regulations in different domains and specific tools and processes used in organizations.* We call for more empirical studies investigating how existing traceability solutions can be applied in practice in different contexts.

Based on the the frequency of the reported challenges and the requirements

Challenge	Studies
Manual work of managing traceability	Paper B, D & E
Diverse artifacts and tools	Paper B & C
Lack of configurable tools	Paper B & C
Unclear traceability processes	Paper F

Table 1.6: Challenges investigated and their corresponding studies.

from our industrial project partners, for Goal 2, the thesis further investigated four challenges from a tool and process perpective. (cf. Table 1.6).

Regarding tool related challenges, we investigated how to minimize the manual work in maintenance of traceability links. In Paper B, which answers **RQ2: What are the primary factors that affect how and to what extent a traceability management tool can provide traceability maintenance?**, we propose important factors that need to be considered by traceability management tools as well as guidelines for traceability tool developers on traceability maintenance. We propose a total of ten guidelines that, when followed, will facilitate more efficient traceability maintenance. In a nutshell, the guidelines propose the following with regards to traceability tools:

- When creating traceability management tools, aim for *flexibility*, both in terms of artifacts, trace link types and work flows supported (G7, G8 and G9) since one tool cannot cover all cases.
- Aim at *maintaining explicit and versioned trace links* as opposed to implicit trace links (G1, G2, G3,G4, G6).
- *Use common technology* that is widely available and/or provides well defined interfaces (G3, G5).
- *Use automation where possible* to maintain the existing set of trace links (G9, G10).

In Paper C, which answers **RQ3: How can a traceability tool be implemented in such a way that it is configurable and extendable?**, we have demonstrated how to apply the proposed guidelines using Eclipse Capra. Eclipse Capra is flexible, offering support to add custom link types and is extendable to support various artifacts types. The tool is a standalone traceability management tool and links created using Eclipse Capra are explicit, meaning that they can be versioned using existing versioning tools. Additionally, the tool supports semi-automatic maintenance of trace links.

In Papers D and E we investigated how semi-automated creation of trace links can be improved since according to our findings in Papers A and B, semi-automation has a higher chance of adoption in industry compared to full automated approaches due to several restrictions such as safety standards in the automotive domain that require having correct trace links. We specifically investigated how to improve the vetting process by first exploring what context information is relevant to human analysts and second, exploring how gamification can improve the vetting process. These two studies contribute to answering

RQ 4: How can the trace link vetting process be improved?. Concrete take aways that we get from these studies are as follows:

- When vetting candidate trace links, the human analyst needs information from the connected artifacts, the traceability information model and the automation algorithm used.
- It is important to carefully choose which role should vet traceability links. Having experience with the artifacts involved increases the chances of making the right decisions.
- While context information is perceived useful, it is more important for the analyst to investigate the actual content of the connected trace links in order to make correct decisions
- The traceability vetting task is a tedious one. Gamification can increase the perceived enjoyment during this task without decreasing the performance of the human analyst.

Lastly this thesis investigates how to introduce traceability that is tailored to a specific organization. This is investigated in Paper F, which answers **RQ5: How can a traceability strategy be established to achieve the specific goals of an organization?**. In this study we propose a methodology to systematically introduce traceability in companies (TracIMo). The methodology has concrete steps and activities that should be performed in each step. In this study, the main takeaways are as follows:

- Always create a traceability strategy based on traceability goals that are *measurable*
- Constantly *evaluate* the traceability strategy to identify needed changes based on changes in traceability goals, tools, development processes and workflows in the company.

Thus, TracIMo calls for using GQM (Goal/Question/Metrics) which allows defining traceability goals as well as the corresponding metrics to track how well the goal has been achieved. Additionally, TracIMo is iterative and therefore allows for modification of the defined traceability strategy until desirable outcomes are achieved.

1.10 Future Work

In this section we discuss potential future work to extend the work presented in this thesis. The overall goal of traceability research is to make traceability management as efficient and effective as possible [85]. We therefore propose the following research directions

Efficient traceability maintenance: In Paper B, we have proposed ten guidelines for traceability tool developers. It is important to be able to link the different guidelines to how they contribute to the efficiency of a traceability management tool w.r.t. traceability maintenance. We propose further studies to investigate how following the guidelines improves

traceability management tools. These studies could also investigate the consequences of not following such guidelines by surveying existing traceability tools which do not follow the guidelines to identify challenges these tools face regarding traceability maintenance.

Transfer of automated approaches to industry: During the thesis, we have had the opportunity to collaborate with various industrial practitioners. Most of the practitioners agree that manual creation and maintenance of trace links is tedious and automation or at least semi-automated approaches are useful. We therefore propose studies that investigate how the various suggested automated approaches in literature can be transferred into practice. The studies could show feasibility of automated approaches in industry as well as propose best practices such as those proposed by Cleland-Huang et al. [28] but geared towards different contexts. For instance one could investigate how to select an automation technique, and how to set up the semi-automated pipeline of creating and maintaining trace links using case studies.

Measurement of traceability: A famous quote by management expert Peter Drucker says "if you can't measure it, you cannot improve it". We believe this is true for traceability as well. To ensure that a certain traceability strategy works, it is important to be able to measure it. While it is already possible to measure things like completeness, the actual usefulness of traceability is difficult to measure and justify [30, 86]. In paper F, we proposed TracIMo, a traceability introduction methodology which uses GQM to define traceability goals and therefore ensures that the goals are measurable. While this is a first step towards measurable traceability, further research is needed to elicit common traceability goals as well as their corresponding potential metrics. Such research could lead to a traceability specific metrics catalog that practitioners can use as a starting point to design measurement systems for their specific traceability strategies.

While these three areas of future work directly follow the studies presented in this thesis, there are several areas we observed that this thesis does not address, but require further research. One of these areas is research on traceability semantics. Traceability semantics refers to the definition of trace link types and their meaning. The artifact types and the link types are usually defined in a traceability information model. One of the steps in TracIMo is the definition of a traceability information model. This model needs to include all link types required to fulfill the traceability goals as well as support existing development processes. The definition of a traceability information model is not a simple task and while there is existing research presenting some examples of traceability information models specific for different tasks (e.g., [87] [88]) there are no specific guidelines on how to define traceability information model in companies. The work by Ramesh et al. [7] and Mäder et al. [89] takes a first step towards this area, but we believe more empirical studies are needed to define best practices and guidelines for managing traceability information models, especially for large scale system engineering companies. Our current work with industry collaborators investigates existing challenges practitioners face when designing

and managing traceability information models and proposes solutions for these challenges.

1.11 Conclusion

While it is agreed that traceability is important, several challenges still remain [19]. This thesis focuses on creating solutions for four challenges from a tool perspective and a process perspective.

As noted by Gotel et al. [85] in the definition of a roadmap for traceability research, there is a need for an updated description of the state of practice of traceability. We first address this gap by providing a detailed description of the state of the art and state of practice of traceability challenges and solutions and analyze how the existing challenges and solutions manifest in practice. Our results show that while there are several traceability solutions for the existing challenges, the applicability of the solutions due to practical constraints is still limited and we conclude that further empirical studies on the applicability of e.g, automated approaches in different contexts are needed.

The thesis further provides solutions for four challenges: (i) high manual work involved in traceability management; (ii) lack of configurable tools; (iii) diverse artifacts and tools used in software development; and (iv) unclear traceability processes. While we investigated the challenges from two distinct perspectives, process and tool perspective, the two perspectives are entangled and tend to depend on each other. For instance, on the one hand, carefully designing how the traceability activities will be carried out needs to take into account the type of tools already used in the organization and on the other hand designing traceability tools need to take into account how the tool will support the different traceability activities.

With respect to the manual work involved in creating and maintaining trace links, the thesis provides a set of important factors and guidelines for maintenance aimed at both traceability tool developers and companies acquiring traceability tools. These guidelines ensure that traceability management tools support traceability maintenance efficiently and therefore reduce the work needed to manually maintain trace links. Additionally, we provide a proof of concept of how to apply these guidelines as well as demonstrate how a traceability management solution can be designed to be configurable and support diverse artifacts and tools. To incorporate automation in the creation of trace links, we see the importance of having a semi-automated approach where a human analyst vets the generated candidate trace links. This thesis sheds some light on the vetting process by investigating how the vetting process can be improved in two perspectives; studying which context information is useful for the human analyst during the vetting process, and investigating the impact of gamification on the task.

To enable the successful alignment of the traceability activities and traceability tools, a carefully planned and and thought through traceability strategy is required. In this thesis we identified that companies either struggle with defining traceability strategies or let the traceability strategies emerge in an ad-hoc manner therefore leading to ineffective traceability management that is expensive and not necessarily beneficial [82]. The thesis proposes TracIMo,

a traceability introduction methodology that consists of explicit steps and activities companies need to take in order to define and deploy traceability strategies. TracIMo is goal-oriented and therefore calls for identifying specific goals of traceability for a company/project and defining a strategy towards achieving those goals. TracIMo also insists on creating measurable goals to facilitate understanding of how the goals are achieved and evaluation of the existing traceability strategy in order to make improvements.

The results of the thesis can be used by both researches and industrial practitioners. From a researchers' perspective, we envision that the provided state of the art and state of practice as well as the mismatch between challenges and solutions can guide other researchers on which challenges of traceability still need to be addressed. From a tool perspective, Eclipse Capra is an open source tool that can be extended and used by other researchers who wish to investigate different traceability concepts. The two studies on the trace link vetting process shed light on how the vetting process can be improved but leave a lot of open questions which could further be investigated in research projects. Lastly, while we have investigated TracIMo in one case study, we believe that the methodology can be used by researchers as a systematic way to conduct other traceability related case studies in industry.

From a practitioners perspective, we envision that the state of the art as well as state of practice described can help them learn about traceability and existing solutions. The guidelines for traceability maintenance can be applied by both companies developing traceability tools as well as companies aiming to acquire traceability tools. The tool, Eclipse Capra can be customized and used by companies who wish to acquire an open source traceability solution. The studies on the vetting process show the importance of choosing the right role when vetting trace links and this information is useful for practitioners when they want to adopt semi-automatic techniques in their traceability practices. Lastly, TracIMo can be used to systematically create, introduce and evaluate traceability strategies in development organizations.

Chapter 2

Paper A

Software traceability in the automotive domain: Challenges and solutions

S. Maro, J.-P. Steghöfer, M. Staron

Journal of Systems and Software 141 (2018): 85-110

Abstract

In the automotive domain, the development of all safety-critical systems has to comply with safety standards such as ISO 26262. These standards require established traceability, the ability to relate artifacts created during development of a system, to ensure resulting systems are well-tested and therefore safe. This paper contrasts general traceability challenges and solutions with those specific to the automotive domain, and investigates how they manifest in practice. We combine three data sources: a tertiary literature review to identify general challenges and solutions; a case study with an automotive supplier as validation for how the challenges and solutions are experienced in practice; and a multi-vocal literature review to identify challenges and solutions specific to the automotive domain. We found 22 challenges and 16 unique solutions in the reviews. 17 challenges were identified in the case study; six remain unsolved. We discuss challenges and solutions from the perspectives of academia, tool vendors, consultants and users, and identify differences between scientific and "grey" literature. We discuss why challenges remain unsolved and propose solutions. Our findings indicate that there is a significant overlap between general traceability challenges and those in the automotive domain but that they are experienced differently.

2.1 Introduction

Over the past 20 years, the automotive domain has witnessed a tremendous increase of software deployed in cars. In today's modern car, software constitutes up to 40% of the production cost [90]. With upcoming trends such as autonomous driving, the software is not only getting more complex but also controls more and more safety-critical functions. The type of software has also shifted from small isolated functions to systems that contain several functions that interact and depend on each other [91]. Such complex systems can cause life threatening accidents when not properly specified, designed, implemented and tested. The number of artifacts produced during development (e.g., requirements, design models, behaviour models, simulations and tests) is large and their creation is usually distributed over various teams, including teams from different companies due to OEM-supplier relationships. With regards to the size of the systems, a typical high-end car consists of features that amount to about 100 million lines of code. This is a very large number as software in other domains has much less lines of code. For example, the F-22 fighter jet has about two million lines of code and the Boeing 787 has around 14 million lines of code [92]. In addition, not only is there a tremendous number of lines of code in this domain but also a large number of other artifacts. For instance, the specifications of the systems in a 2004 car had already reached 20000 pages at that time [1]. This can be overwhelming if there are no standardized methods established to keep track of these artifacts, their relationships, and how they evolve.

In such situations, traceability plays an important role. In this paper, we define traceability as the ability to relate artifacts created during the development of a system, thus following [5]. Traceability helps in understanding which artifacts are connected to each other and allows to keep track of which features have already been specified, implemented and tested. Traceability plays an even bigger role for maintenance tasks by facilitating change impact analysis and improving understandability of the system for the developer who needs to make changes in the system [85, 93]. In the automotive industry, these aspects are particularly important in light of safety standards that require proof that safety requirements were specified, taken into account during the design and development, validated in test cases, and verified through safety analysis (see, e.g., [94, 95]).

In order to realize the benefits of traceability (and successfully argue their safety cases), software development companies need to establish a traceability strategy that is aligned with their goals. Defining and implementing a traceability strategy is not a trivial task, since it requires a good understanding of the artifacts to be traced as well as the ability to define meaningful links and to make sure the created links are useful [10].

On the one hand, there exists a large body of knowledge on traceability; for instance between 1999 and 2012, 70 studies on traceability were published in just the Requirements Engineering (RE) conference [61]. On the other hand, in practice traceability is either not established at all [96] or only established since standards demand it [38]. In addition, it is unclear how the traceability challenges in the automotive domain relate to general traceability challenges and how they manifest in a practical environment of a company.

The contribution of this paper is therefore to provide an exhaustive empirical evaluation of traceability challenges and solutions in the automotive domain that takes the specific characteristics of automotive software development into account. To achieve this, we conducted a tertiary literature review, a case study, and a multi-vocal literature review. This allows us to explore the traceability problem in the automotive domain from both the practical and the scientific perspective and provides insight into the challenges of traceability as they present themselves in practice as well as solution approaches proposed by academia, tool vendors, consultants, and the users of traceability themselves. Our aim is to give insights on which challenges exist in this domain, the spectrum of solutions available, and highlight difficulties experienced with using some of these solutions in the automotive domain. Our study therefore investigated the following research questions:

RQ 1: What are the general traceability challenges and solutions reported in literature?

RQ 2: What are the particular traceability challenges and solutions in the automotive domain?

RQ 3: Which of the reported traceability challenges in scientific literature and non-scientific literature can be observed in practice in the automotive domain and how have they been solved?

To obtain data for our study, we used three different data sources: a tertiary literature review in which we reviewed 24 secondary publications on traceability; a case study at an automotive supplier company; and a multi-vocal literature review in which we reviewed a total of 245 scientific and non-scientific sources. We found 22 challenges from the literature of which 17 were also found in our case company. Five of the challenges have been solved with solutions proposed in literature, six are partially solved while six remain unsolved even though there are proposed solutions in literature.

This paper extends our work reported in [97] in which we discussed challenges related to creation, maintenance and exchange of traceability by also discussing traceability challenges related to preparation and planning for traceability and the use and measurements of traceability. We have also added the multi-vocal literature review as an additional data source and extend our results and discussion with this new trove of information. Furthermore, we review persisting challenges in detail and give an overview for solutions viable in the automotive domain.

The rest of the paper is structured as follows: Section 2.2 describes traceability requirements in the automotive domain and our research method is described in detail in Section 2.3. Sections 2.4 to 2.7 present the challenges and solutions and describes them from the perspective of the tertiary and multi-vocal literature and how they relate to the case company. Section 2.8 provides a discussion of the results. Limitations of the study are discussed in Section 2.9, Section 2.10 discusses related work, and Section 2.11 concludes the paper and outlines future work.

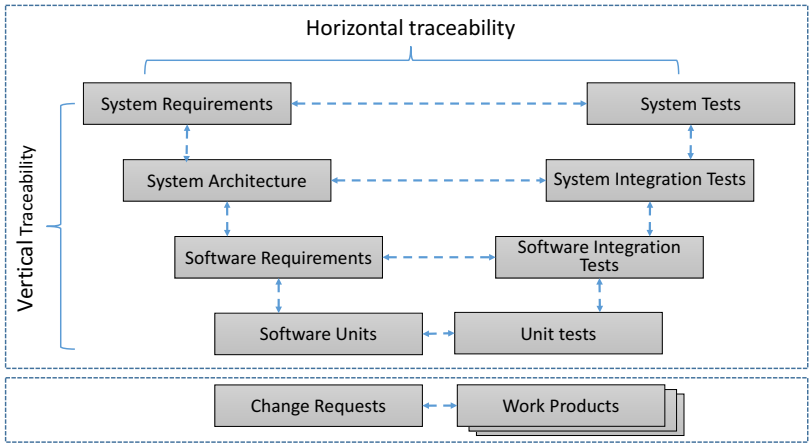


Figure 2.1: Traceability in the V-model

2.2 Traceability Requirements in the Automotive Domain

In this domain, traceability requirements are imposed by the ISO 26262 [75] – a functional safety standard for road vehicles – and ASPICE [65] – a process assessment model specific to the automotive domain. Both the ISO 26262 and ASPICE prescribe the use of a V-model process lifecycle for *product* development of embedded systems. The traceability links required are shown as dotted lines in Figure 2.2. It is important to note that due to the overlap in ASPICE and ISO26262 [98], these standards are usually used in companies to complement each other, rather than as two separate alternatives. Since ASPICE is a process assessment model, it can, e.g., be used to assess the maturity of a process with extensions that also cover the safety critical aspects prescribed in ISO26262 [99]. In summary, both standards impose the following with respect to traceability:

Vertical traceability: Artifacts must be traceable to their children and the children should be traceable to their parents (*bi-directional traceability*). An example of this is that a requirement should be traceable to architectural artifacts (structural and behavioral) that realise it and to the code associated with these artifacts. It should also be possible to trace from the code to architectural artifacts and back to the requirement.

Horizontal traceability: This means that it should be possible to trace from artifacts on the left side of the V-model to their verification artifacts (such as tests or safety analysis) on the right side of the V-model. In addition, traceability links should be created and maintained between any recorded change requests and the work products affected by them to enable change impact analysis.

From a traceability point of view, the main difference between the two is that ISO 26262 requires traceability to be established between safety-related

artifacts, i.e., it requires defining links from hazards to safety goals, to safety requirements, to the structure and behavior of these safety requirements, to the code and to tests that are responsible for testing all the safety artifacts. ASPICE on the other hand requires traceability for *all* artifacts, even if they are not safety-related. Another difference is that while the ISO 26262 standard *recommends* bi-directional traceability, it is deemed mandatory in ASPICE. The term “recommended" in ISO 26262 implies that companies are free to choose other alternatives to show that the requirements have been fulfilled. Additionally, the ISO 26262 standard requires the artifacts to be versioned and have unique identifiers in order to be traceable.

Apart from the requirements imposed by the standards, traceability processes and tools in this domain need to deal with characteristics such as complexity, longevity, and variability of the products as well as the distributed development environment. In this situation, traceability can, e.g., help with program comprehension [100], to allow change impact analysis [101], and to document rationale and design procedures [86]

2.3 Research Method

The aim of our study is to get an understanding of the traceability problem in the automotive domain and thus to answer the following research questions:

RQ 1: What are the general traceability challenges and solutions reported in literature?

RQ 2: What are the particular traceability challenges and solutions the automotive domain?

- **RQ 2a:** What are the challenges and solutions regarding traceability when addressing the demands of automotive standards ASPICE and ISO 26262?
- **RQ 2b:** What are additional relevant traceability challenges and practices in the automotive industry?

RQ 3: Which of the reported traceability challenges in scientific literature and elsewhere can be observed in practice in the automotive domain and how have they been solved?

To answer these research questions, we collected data from general and specific scientific literature, from a case study and from specific non-scientific literature. To achieve this, we used three types of research methods: a tertiary literature review, a case study with an automotive supplier, and a multi-vocal literature review. The tertiary literature review provided data on the challenges and solutions in the literature (RQ 1). The multi-vocal literature review allowed us to include information about challenges and solutions that were not reported in scientific literature and were thus not covered in the tertiary literature review (RQ 2a and RQ 2b, RQ 3). Since the adoption of standards in the automotive domain is evolving quickly, RQ 2a addressed this topic specifically. We regarded RQ 2b as an auxiliary research question, designed mainly to provide us with additional material that does not mention the automotive standards. The case

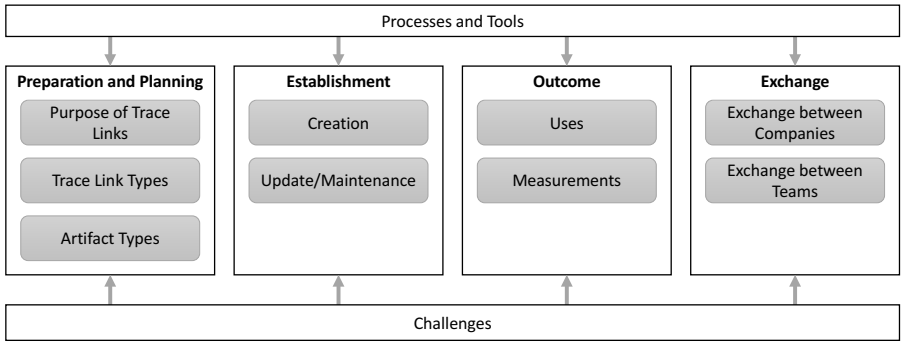


Figure 2.2: The scope of the case study

study provided data on which challenges exist in practice and their solutions if any (RQ 3).

2.3.1 General Guidelines and Scope

We conducted the tertiary literature review according to the guidelines in [102], the case study according to the guidelines proposed in [64] and the multi-vocal literature review according to the guidelines proposed in [103]. Before conducting these studies, we defined the scope relevant to us and which all three data sources cover. Our scope (depicted in Figure 2.2) indicates that we distinguish four different traceability categories (*Preparation and Planning*, *Establishment*, *Outcome* and *Exchange*) which are inspired by the generic traceability process model defined by Gotel et al. [10]. We used this model because it contains most of the activities needed for establishing traceability. This model is also well-known in the traceability community and since, its definition has been used in other research, (e.g., in [34, 55, 104]) as a basis for understanding and describing traceability.

In the model, the *Preparation and Planning* category, focuses on the processes and tools involved when preparing to include traceability in a company or a particular project. The *Establishment* category deals with the processes and tools involved in the actual creation and maintenance of traceability links. The *Outcome* category focuses on how the links are stored and how they are actually used after they have been established. Since we are studying the automotive domain where the OEM-Supplier relationship means that artifacts are exchanged between companies, we added a fourth category called *Exchange* where we discuss challenges of exchanging traceability within and between organizations.

The details of the tertiary literature review are described in Section 2.3.2, those of the multi-vocal literature review in Section 2.3.3, and those of the case study are described in Section 2.3.4. The entire research process is summarized in Figure 2.3.

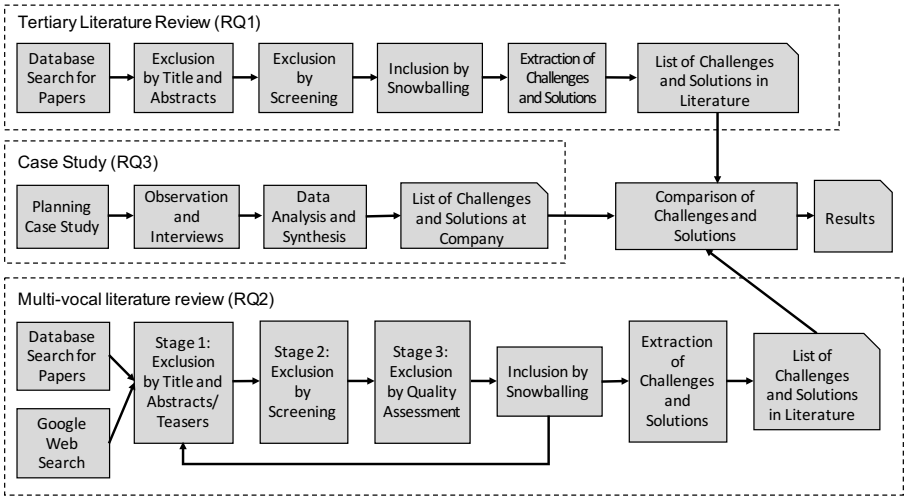


Figure 2.3: Summary of the Research Method

2.3.2 Tertiary literature review

Our tertiary literature review followed the guidelines for conducting a systematic mapping study as proposed by [102]. The guidelines indicate that a systematic literature study should include five steps which are *Definition of research questions*, *Conduct search*, *screening of papers*, *Keywording using abstracts* and *Data extraction & mapping process*. The subsections below describe how these steps were carried out in our study.

2.3.2.1 Definition of Research Questions

Our aim is to identify both general traceability challenges and solutions from the literature study that we can later compare to the specific challenges and solutions in the automotive domain from the multi-vocal literature review and the case study. Therefore our literature study has to answer the following research question:

RQ 1: What are the general traceability challenges and solutions reported in literature?

2.3.2.2 Conducting the Search

Since this is a tertiary literature review, our aim was to find literature reviews published on traceability in the domain of computer science. We searched three databases : Scopus, ACM Guide and IEEE Xplore. The search strings used are shown in Table 2.1. This search led to a total of 522 papers which were reduced to 370 by removing duplicates.

2.3.2.3 Screening of Papers

By reading the title and abstract, we selected papers that are relevant to our study using the following inclusion criteria:.

[a] The paper reviews literature on traceability.

[b] The paper is published in a peer-reviewed venue.

[c] The paper is in the field of computer science.

[d] The paper mentions challenges and solutions of traceability and gives a description of these challenges and solutions.

[e] The paper is in English or German.

The initial screening in which we read the title and abstract left us with 27 relevant papers. After this we further read the introduction and conclusion of the papers and excluded eleven more papers because they did not fulfill criteria number one or four. From the remaining 13 papers we used both forward and backward snowballing to look for papers that specifically addressed challenges of traceability. We limited our snowballing to papers published between 2007 and 2017 to ensure that we get current traceability challenges. This led to an addition of eleven more papers. In the end, we had identified a set of 24 relevant papers.

2.3.2.4 Data Extraction and Classification

We examined all 24 papers, extracted all the challenges and solutions they report and listed them in a spreadsheet. After this process, we reviewed all the challenges in the list and placed each challenge in the best-fitting sub-category in the conceptual model shown in Figure 2.2. At this stage, we observed that some of the challenges could be placed in more than one sub-category (for instance, the challenge of *Manual work* could be placed in both the creation and maintenance sub-category). We therefore merged these sub-categories. Afterwards we reviewed the challenges and discovered that they could be further distinguished by challenges about technical issues in particular with the tool support, human factors that involved employees, and the organisational setting and established processes. Therefore we divided the merged category of creation and maintenance into three sub-categories: tool support, human factors and organisation and processes. Additionally, in the *Preparation and Planning* category, all of the challenges found were related to the general understanding of traceability. We therefore merged the three sub-categories (Purpose of trace links, trace link types and artifact types) into one category "knowledge of traceability". It also became clear that the distinction between company-internal and external *Exchange of Traceability* was not helpful, in particular since a *Lack of coordination* could be found in both sub-categories. The *Outcome* category remained the same as in our conceptual model.

2.3.3 Multi-vocal Literature Review

To answer the sub-research questions on practices for traceability in the automotive domain (RQ 2a and RQ 2b), we use a multi-vocal literature review (MLR) [103]. This strategy allows us to include non-academic sources (sometimes called "grey literature"). Since many practitioners do not publish their experiences in scientific venues, but do publish whitepapers, write blog entries,

or give presentations at trade shows and conferences, this allows us to better grasp the current state of the art and practice. Due to the large number of sources, this approach requires very strict selection criteria for the sources. We have detailed these criteria below. We used SCOPUS, IEEE Xplore and ACM Guide as sources for scientific literature and Google Web Search to find non-academic sources.

We used the general protocol suggested in [105] for the MLR. It contains three stages of evaluation and combines systematic and "opportunistic" discovery. While the former form of discovery is covered by the searches in scientific databases and Google Web Search, the latter includes papers that were recommended by colleagues, the results of snowballing, or discoveries of sources that were otherwise incidental and not part of the systematic search. The evaluation is based on the source title, teaser text, and quickly following the link to check the source (Stage 1), the entire text (Stage 2), and an assessment of the overall quality of the source (Stage 3).

2.3.3.1 Definition of Research Questions

Our aim is to find traceability practices relating to standards compliance in the automotive industry and challenges and solutions to traceability that are not reported by the scientific literature. We therefore use the MLR method to answer the following research questions:

- RQ 2a: What are the challenges and solutions regarding traceability when addressing the demands of automotive standards ASPICE and ISO 26262?
- RQ 2b: What are additional relevant traceability challenges and practices in the automotive industry?

The data collected in the MLR will also support our answers to *RQ 3: Which of the traceability challenges reported in scientific literature and non-scientific literature are also evident in practice in the automotive domain and how have they been solved?*

2.3.3.2 Conducting the Search

As mentioned above, we use Google Web Search to look for non-scientific sources. In order to keep the number of results manageable, we let Google filter redundant entries. The number of results Google reports on the first page of the search results indicates the total number of hits in the index. The final number (after filtering) becomes evident when going through the search results page by page and navigating to the final results page[1]. We used Google Search for our campus location (Gothenburg, Sweden) in early-to-mid August 2017. The search for RQ 2a was conducted with a fresh user profile without any cookies or history in Google Chrome to avoid influencing the results through tracking cookies. The search for RQ 2b was conducted with a simple command line script, again avoiding any tracking. Searches for RQ 2a and RQ 2b have been

[1]For a more technical discussion of this filtering, please refer to `https://support.google.com/gsa/answer/6329272`. In essence, Google filters duplicates and only displays the two most relevant similar pages on the same domain.

performed with different IP addresses to also avoid tracking of this information. The search terms have been intentionally kept vague to increase the breadth of the found sources, even if this increased the effort for the screening of the sources.

To answer **RQ 2a**, we combined the two pre-dominant process standards Automotive SPICE and ISO 26262 with the term traceability. Google Web Search reported approx. 34700 results in its index. When browsing through the result pages, the total number of hits is 263. We exported the search results to a spreadsheet and enriched it with meta-data as described below. The number of search results for scientific sources was significantly smaller, with a large overlap between the data bases. The exact search terms and result numbers are shown in Table 2.1:

To answer **RQ 2b**, we used a very general core search term ("automotive traceability") that we paired with exclusion terms to reduce the number of hits (cf. Table 2.1). For the Google Web Search, we also included German search terms to look for information provided by the sizable German automotive industry. This was not done for scientific literature since the used databases focus on English language literature. As there is no unique German translation for "traceability", we use different options that have been used in literature based on a preliminary search: "Nachverfolgbarkeit", "RÃijckverfolgbarkeit", and "Verfolgbarkeit". Unfortunately, these terms are not only used in the context of software, but also in the context of being able to trace work pieces in a manufacturing process or products in a supply chain. We thus decided to refine the search term to exclude these irrelevant results. Based on a preliminary sighting of results, we excluded the terms "Manufacturing", "Food", "Laser", "Barcode", "supply chain", "logistics chain", "lot tracking" "ChargenrÃijckverfolgung", and "rfid". This increased the relevance of the search results significantly. Since we were interested in traceability challenges and practices that are currently relevant, we reduced the relevant time span for scientific literature to 2008 to 2018.

As for the data for RQ2a, we exported the search results to a spreadsheet and enriched them with meta-data. To check if there were additional reports by automotive OEMs that we did not pick up through this search, we also searched for combinations of OEM names with traceability and its German counterparts. No additional sources were found this way.

2.3.3.3 Screening of Sources

We screened the papers in three stages, similar to the way described in [105]. In Stage 1, we excluded sources based on the title of the page and the description by Google Web Search and followed the link to determine if the information was accurate, which type of source we dealt with, and if the source was available. In Stage 2, we regarded all available information about the source, i.e., its full text and its meta-data (e.g., which website the source was found on, the authors if they were identifiable). In Stage 3, we evaluated the quality of the source, successively excluding sources that did not provide useful information. This exclusion was mostly focused on deciding whether criteria 1 to 6 below were met. Since the process was highly context-dependent, it is difficult to describe the exact criteria used. Possible exclusion candidates were therefore marked by

one researcher and then confirmed (or overruled) by another in order to ensure unbiased results. The inclusion criteria for stages 1 and 2 were as follows:

[a] The source provides first-hand information and is not an encyclopedic article (such as Wikipedia)

[b] The source needs to be publicly available (i.e., not behind a paywall inaccessible by researchers or only available on request)

[c] The source is in written format (e.g., not a video or an audio file)

[d] The source is written in English or German

[e] The source has not been considered previously (to avoid duplicates)

[f] The source was not written by the authors of this paper (to avoid all sources that are directly related to our own research and could therefore be perceived as biased)

The inclusion criteria for stages 2 and 3 were different depending on the research question:

[a] (RQ 2a) The source discusses approaches to traceability based on the standards and does not only mention them as motivation.

[b] (RQ 2a) The source provides information about traceability concepts in the context of the standards (and does not, e.g., only advertise functionality of a tool [2] or describe the need for traceability).

[c] (RQ 2a) The source refers to important standards in the automotive industry.

[d] (RQ 2b) The source discusses traceability practices in the automotive industry.

[e] (RQ 2b) The paper discusses traceability in a software development process (as compared to, e.g., traceability of the origin of parts used in the manufacturing of a vehicle).

2.3.3.4 Data Extraction and Classification

As part of the enrichment with meta-data, we identified the provenance and type of the source. Possible values for provenance – describing to which group of people the authors of the source belong – were academia, tool vendor, consultant, user, standardisation body, agency, student, Open Source Community, mixed, and unknown. As a source for this information, we used the stated affiliations, the website the source was found on, or meta-data provided with the source. In case the provenance was not obvious from the source directly, author names were identified and used to search for their affiliations. In case the affiliation of the authors was with a company that fit several of the possible values for

[2] While we acknowledge that tools play an important part for traceability, pure marketing material does not describe practical applications and uses.

provenance (e.g., a tool vendor that also engages in consultancy services), we selected the one that fit the type of the source best.

We identified a large number of different source types. Among the most prominent were whitepaper, presentation, tool documentation, blog entry, job posting, course announcement, tool description, and manual. An overview of the frequency of provenance and source types is given in 2.11.

There is overlap between the search results for RQ 2a and RQ 2b. Of the 659 sources totally regarded for RQ 2b, 78 were already analysed for RQ 2a. The number of sources in German is relatively low with a total of 33 out of the 659 considered. Of these 33, 17 were considered relevant and are included in the analysis. Overall, 125 and 120 unique sources were considered for RQ 2a and RQ 2b, respectively.

The data collected for both sub-RQs has been analysed together to answer the over-arching RQ 2. The overlap between the found sources and the relative semantic proximity of the search terms makes a joint analysis prudent. The same codes have been used in both cases to identify challenges and solutions as well. Two researchers engaged in this activity. The starting point for coding challenges were the codes identified in the tertiary literature review (cf. Section 2.3.2). Codes for solutions were emergent and refined in several rounds of discussions between the researchers. Cross-checking was performed and edge cases were reviewed and discussed. Sources excluded as off-topic or containing no challenges or solutions by one researcher were reviewed by a second researcher to avoid accidental exclusion of relevant material.

2.3.4 Case Study Design

As previously mentioned, the case study followed the guidelines reported in [64]. The aim of the case study was to provide empirical evidence of the challenges and solutions found in the literature, show how these challenges manifest in practice and identify new challenges that were not reported in literature. Furthermore, the study provided context for the challenges and solutions found in both the tertiary review and the multi-vocal review and therefore provided data to answer *RQ 3: Which of the reported traceability challenges in scientific literature and non-scientific literature can be observed in practice in the automotive domain and how have they been solved?*

2.3.4.1 Case and Subject Selection

The study was conducted in one of the world's largest suppliers of automotive components located in Germany. The company is multi-national which means that development is distributed in various locations. The company develops various types of automotive components ranging from hardware-only components to software-only components to embedded systems which include a software deployed on a certain hardware component. For this study we were interested in traceability during automotive embedded systems development.

Our case study has two units of analysis within the same company: two departments both developing embedded systems at the company. Since our aim was to investigate how traceability challenges manifest in practice, we selected these two departments because they already implement traceability in their

projects and develop safety-critical embedded systems for which traceability is a mandatory requirement. The two departments were also interested in improving their traceability practices, thus the topic was relevant and of interest to them. To be able to understand how traceability is implemented throughout the development life cycle, we conducted the study with seven participants in the following roles: two senior experts working on traceability (one from each department), four software system architects (two from each department) and one functional developer who belongs to one of the departments. We selected these roles in order to get a full picture on how development is done from when a requirement is received to when it is implemented and tested. The first role of senior expert is responsible for understanding what traceability needs the department has, surveying feasible solutions, acquiring these solutions and making sure that they are used in the department. The second role, system architect, is responsible for receiving requirements from the customer, breaking them down and assigning them to development teams. This role is also responsible for managing the architecture of the systems that the department is developing. The last role, developer, is responsible for implementing the features and testing them. In one department, the role of developer and tester are split into two separate roles assumed by separate people.

2.3.4.2 Data collection procedure

We collected data through observing demonstrations and conducting semi-structured interviews. Observations enabled us to understand the development process and how traceability activities are carried out and the semi-structured interviews enabled us to gather comparable data on the challenges. The model describing the scope of our study and interview questions were sent to the participants a week before the study took place. This was to allow them time to prepare for the demonstrations and interviews. For each participant, we started with the participant giving a demonstration on how they implement traceability using the scope model as a guide. This was followed by a semi-structured interview. The interviewer only asked questions which were not answered by the demonstration part. Due to legal issues, the interviews were not recorded but the interviewer took notes. The interviews and observation for each person lasted between 90 minutes to four hours with breaks in between. The longer sessions were with senior experts who explained and demonstrated the traceability process in detail. The interview guide for these interviews is available online[3].

2.3.4.3 Analysis procedure

The data analysis started immediately after the observations and interviews were completed. This was to ensure that all relevant information was recorded for later analysis since the interviews were not recorded for legal reasons. The interviewer drafted a summary of the sessions and what was learned from the study and presented it to one of the senior experts for confirmation. During this presentation, the interviewer described the development process and outlined the challenges that were learned from the interview. The senior

[3] `https://tinyurl.com/ycjrqal4`

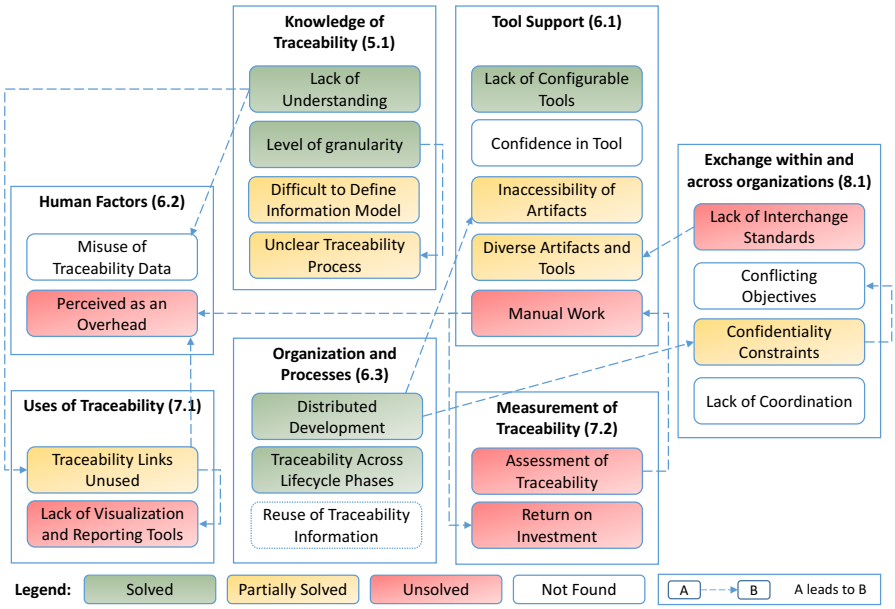


Figure 2.4: Summary of Traceability Challenges. The solved challenges have a green background, the partially solved challenges have a yellow background and the unsolved challenges have a red background. The challenges that have no background color were only in the literature and not identified in the case study. This means that the data collected was not sufficient to say if these challenges exist in the company. The directed arrows mean that one challenge leads to the presence of another challenge.

expert could then confirm the findings or correct the findings when things were misinterpreted by the interviewer. The senior expert could also ask questions at any time during the presentation. This exercise led to few changes, indicating that most of the initially collected information was correct. After this, we went through the interview notes and identified the challenges. We used the categories in the interview model as analysis codes and placed each challenge found in the appropriate category.

2.3.5 Results

In the next sections (Section 2.4 to 2.7) we report findings both from the tertiary literature review, the multi-vocal literature review, and the case study separated by the categories we also used to scope our case study (cf. Figure 2.2). The challenges are summarized in Figure 2.4. In the figure, we also include the relationships between the challenges we discovered during analysis. The arrows is indicate the dependencies between challenges: if one is present then the other is also likely to be present. We describe each challenge, discuss it and its solutions in the context of both the tertiary literate review and the multi-vocal review and then compare them with the challenges and solutions at the company.

2.4 Results: Preparation and Planning

This section describes the challenges and solutions that are encountered when companies are preparing to include traceability either in a specific project or the entire company. Such challenges are concerned with the availability and perception of knowledge about traceability by managers, engineers and developers.

2.4.1 Knowledge of Traceability

We found four challenges related to knowledge about traceability in the literature. All four were also found at the company. Two of the challenges have been solved while two have only been partially solved using work-arounds.

2.4.1.1 Lack of Knowledge about and Understanding of Traceability

Description: In order to prepare and plan for traceability in a company, both the managers and developers need to have an understanding of what traceability is and its purpose. This understanding also needs to be aligned, meaning that all the people in the company should have a common interpretation of what traceability is. For companies, if the concept of traceability is not clear, then the chances of failure are high.

Challenge and its Solutions in Secondary Literature: This challenge has been reported by nine papers from our tertiary review [8, 34, 38, 61, 106–110]. In [34], for instance, the authors report that some companies, especially those not working in a safety-critical domain, have no notion of the term traceability. Another issue is that different individuals in the company have a different understanding of the purpose of traceability [8]. The most common is that managers see it as a mandatory task that needs to be done for certification purposes while developers perceive it as bureaucracy and a waste of time [38, 108]. In some cases where traceability tools are well-established, developers may perceive it as important and useful for tasks such as impact analysis [107]. The literature proposes that in order to achieve a common understanding of traceability among all stakeholders, training is important. Early on, the company should invest some time and effort to train its employees on purposes and practices of traceability. The training should also discuss semantics of traceability links, completeness, traceability link quality, and other topics [108].

Challenge and its Solutions in the Automotive Domain: This challenge was reported 13 times in the the multi-vocal review. In most cases, it was reported in the context of stakeholders in the automotive domain not understanding what kind of traceability is required by the different standards such as ISO 26262 and ASPICE. This is because the standards do not define how traceability should be established concretely (e.g., [111, 112]). It is also not clear how traceability should evolve in the course of the project (e.g., [113]. Training is also suggested as a solution (e.g., [114] and [115]). Furthermore it has been suggested that certification bodies should provide some guidelines on how to properly establish traceability (e.g., [116, 117]).

Comparison to Case Company: This challenge exists at the company but has already been solved. Given that the company operates in a safety-critical domain, employees are already aware of the concept of traceability. They base their understanding of traceability on the requirements defined by the safety standard they need to comply to (ASPICE). They even have expert roles whose job is to understand what the standards require, form a strategy on what they need to do to comply, and communicate this to the rest of the company.

2.4.1.2 Difficult to Define Information Model for Traceability

Description: Traceability links can be of different types depending on their purpose and which artifacts they connect. The link types can differ from domain to domain. Traceability link types are usually defined in what is known as a traceability information model. It can, e.g., take the form of a meta-model, a database schema, or an ontology. Link types can be generic and carry little or no semantics (for instance a link type called "related_to" that allows connecting arbitrary artifacts) or they can be specific and carry meaningful semantics (for instance a link type named "tested_by" that can only connect a requirement and a test in the sense that the requirement is tested by the connected test). Defining traceability links with domain specific semantics is advantageous as it allows for analysis of the links based on the semantics. In order to define the traceability information model, one needs to understand which types are needed and useful in the specific domain, company, or even project. These needs can evolve over time as well. That makes it difficult to reuse existing information models and to settle on an information model that will remain fit-for-purpose over a longer period of time.

Challenge and its Solutions in Literature: This challenge was reported by seven of the papers [5, 8, 22, 34, 109, 118, 119]. One of the solutions proposed is to define a standard traceability information model [5] after observations in various companies. This model can indeed be used as a starting point for companies to define their traceability metamodel. However, since this is a domain-specific problem, another solution proposed is to document domain-specific guidelines on how to define metamodels [8]. This can be done through reporting case studies or experience reports.

Challenge and its Solutions in the Automotive Domain: Seven of the sources in the multi-vocal literature report this challenge. Development companies find it difficult to define a traceability information models especially for tracing to non-functional requirements [120] and product line artifacts [121]. In some cases the companies have no traceability information model at all [113]. Solutions to this challenges are to have evolvable semantics [116], which means that the traceability information model can be designed in an iterative manner and evolved until it is sufficient. Another solution is to derive the traceability information model from the development process by analysing the process and the involved artifacts [122].

Comparison to Case Company: At the company, this challenge exists and it is partially solved. In both departments, the traceability metamodel has already been defined following the ASPICE standard (cf. Figure 2.2). However, the links are designed specifically to adhere to this standard and it is

not clear if these link types assist developers in their development activities. The standard also does not have any guidelines for which links should be created, for development paradigms such as product line development. The two architects interviewed reported that the company has product lines with many variants but they do not know how to include traceability links that take into account variability. This is because the traceability information model defined, does not take into account concepts of variability. Another issue that is not addressed in the standard is how to trace to non-functional requirements such as performance and security. The plan in the company is to use the current metamodel and collect data from its users on what is missing or which links are not working in order to evolve the model.

2.4.1.3 Level of Granularity

Description: When designing and planning a traceability concept in a company or even a project the level of granularity on which the traceability links should be created on, has to be defined. For instance, a decision must be made if a requirement should be linked to a test file, a test case, or a particular line of code in a test case. This is a challenge, because, if the links are too coarse-grained, they do not provide sufficient detail. If they are too fine-grained, however, their number can become overwhelming and confusing to the end users.

Challenge and its Solutions in Literature: This challenge was reported in three studies [110, 118, 123]. The solution suggested is that the granularity of the links should be defined explicitly in the traceability information model and the traceability links should be checked regularly to ensure that the links are created with the right level of granularity. This solution however does not suggest which level of granularity a project or company should use.

Challenge and its Solutions in the Automotive Domain: This challenge was reported by 17 sources from the multi-vocal review. The main problem is that at different phases of the system development (requirements engineering, design, implementation etc.) artifacts are defined at different levels of abstraction. This also makes it difficult to determine which abstraction level is appropriate for linking as the mapping from the different development phases is not one to one. For instance, when linking from requirements to architectural components, one has to decide whether to link to high level components of the system or to detailed classes within the components. The multi-vocal review did not point to any solutions to this challenge except that a lot of experience with traceability is important in order to determine the right level of granularity [124].

Comparison to Case Company: At the company, this was observed as a solved challenge. The company adopted the level of granularity implied by the V-Model as suggested by the ASPICE standard. The system requirements are derived from customer requirements. The system requirements are then broken down into functional requirements which could be software requirements or hardware requirements. Software requirements are further refined into detailed software requirements. The developer is then assigned a detailed software requirement for implementation. Traceability links are created from customer requirements to software requirements to detailed software requirements. The

detailed software requirement is then linked to an implementation file that actually contains the code. The detailed software requirement is also linked to a test.

2.4.1.4 Unclear Traceability Process

Desciption: Establishing a traceability strategy requires a traceability process (how links are created, used, and maintained) to be put in place. Such a traceability process should be aligned with the software development process that already exists in the company. It is important for the traceability process to refer to work products of the existing development process. This process should also define roles and responsibilities regarding traceability in the company. If such a process does not exist or is vaguely defined, links will be created in an ad hoc manner which results in low link quality.

Challenge and its Solutions in Literature: Ten of the papers reviewed report this as a challenge [9, 14, 22, 38, 107, 109, 110, 119, 123, 125]. In [123], the authors propose that the solution is to create a traceability process based on the traceability metamodel defined at the company. This process should be documented and communicated to all stakeholders early on. Managers should be assigned the role of making sure that this process is followed. In [22], the authors propose putting in place an automated process of creating traceability links by generating skeletons of artifacts from requirements and their traceability links and let these skeletons be filled as development goes on.

Challenge and its Solutions in the Automotive Domain: This challenge was reported by 28 sources from the multi-vocal review. In the automotive domain, standards that need to be followed have an impact on how the traceability process should be defined. Sometimes there is a mismatch between the standards' requirements and the process at the company which makes defining the traceability process difficult [116]. Solutions proposed are to use the standards to derive a traceability process [126]. This way the company can be sure of its compliance. Another solution is to make sure that the defined traceability process is enforced in order to avoid traceability tasks being performed too late in the development and in an ad hoc manner (e.g., [117,127]). Moreover, having tool support such as an integrated tool platform where all development activities are done or a structured way of defining artifacts also helps to solve this challenge [128].

Comparison to Case Company: At the company, this challenge exists and has been partially solved. A traceability process already exists and although it is a completely manual process, the developers and architects are aware of which links need to be created based on the breakdown of the requirements as discussed previously. In one department, the requirements are defined as use cases and therefore traceability links are created from use cases to design, implementation and tests. In the other department, the requirements are defined as user stories and therefore the links are created from low-level user stories to design, implementation, and test. This challenge is partially solved as there are currently no roles that can check if the process for creating traceability links was followed. Sometimes during review meetings, flaws of traceability links can be detected and fixed.

2.5 Results: Creation and Maintenance

This section reports on challenges that are associated with the activities of creating and updating the traceability links. The challenges are divided into three categories: tool support, human factors, and organization & processes.

2.5.1 Tool Support

We found five major challenges in the literature which were reported in this category. Four of these challenges were also found at the case company. On further analysis only two of these challenges have been solved, one has a workaround solution, while two of them still remain unsolved.

2.5.1.1 Lack of Configurable Tools

Description: Traceability needs can greatly differ from company to company and even from project to project. Therefore, providing a tool that can only be used in a specific context is a limiting factor. It is crucial for tools to allow for customization in terms of link types, supported artifact formats, reporting, selection of relevant information, etc.

Challenge and its Solutions in Literature: This challenge was reported by six studies in our review [8, 34, 38, 80, 108, 129]. The solution described is urging developers of traceability tools to take into account how flexible the tool should be. For instance traceability tools should be flexible in a sense that they allow definition of custom links, allow linking to arbitrary artifacts, be able to define which reports should be created from the links and so on. The more flexible the tool, the better companies can tailor it to fit their project needs.

Challenge and its Solutions in the Automotive Domain: From the automotive literature, this challenge was reported by 13 sources. Most of these sources report that tools do not support the definition of custom traceability links with rich semantics. The solution suggested for this challenge is similar to the ones from the tertiary review. Traceability tool vendors need to design flexible traceability tools that are highly customizable (e.g., [130]).

Comparison to Case Company: This is one of the challenges that the company has solved. For requirements management, they have adopted DOORS[4], a tool that is flexible and allows for definition of custom traceability links. Out of the box, the tool allows defining custom link types between requirements. Other artifacts that are stored outside the tool can be linked through OSLC[5] (Open Services for Lifeycle Collaboration) which is a standard for sharing artifacts across tools. For artifacts that do not have OSLC representations, special attributes in the requirements can be defined to store IDs or names of artifacts that are outside the tool. While OSLC enables creating links to artifacts in external tools, maintaining consistency of these external links is a challenge as when artifacts evolve in their tools, the changes are not propagated to other tools for the links to be updated accordingly.

[4] http://www.ibm.com/software/products/en/ratidoor
[5] http://open-services.net

2.5.1.2 Confidence in Tool

Description: Development companies need to have confidence in the traceability tools that they acquire. One way to establish this confidence is to use tools that have been certified for specific standards. Such a certification provides evidence that the tool works as expected and does not, e.g., introduce errors in the safety analysis that could lead to an unsafe product. It is also important to make sure that the tool is scalable since large and complex systems with a large number of traceability links are common in such domains.

Challenge and its Solutions in Literature: In the tertiary review, three of the sources report this as a challenge [34], [80], [125]. Two aspects have been discussed: 1) companies have problems finding tools that will enable them to be adhere to the necessary safety standards [125]; 2) companies have no confidence in the scalability of the tools they acquire as they have not been used in large-scale development [34], [80]. There are no concrete solutions suggested for this except that tool vendors should design flexible and scalable tools [8].

Challenge and its Solutions in the Automotive Domain: Only four of the multi-vocal literature report this challenge. In the automotive domain where requirements can be up to 2000 pages, it is unclear whether existing tools will scale to this level ([131]). Furthermore companies have to be sure that the tools they acquire will support them in being compatible to the different safety standards. To address the confidence challenge in terms of adherence to safety standards, tool vendors now provide solutions that are certified for these respective standards already (e.g., Polarion [132] and Jama [133] are both ISO 26262 certified).

Comparison to Case Company: This challenge was not reported at the case company.

2.5.1.3 Inaccessibility of Artifacts

Description: When creating or updating a traceability link, it is crucial to have access to the artifacts that need to be connected by the traceability link. In a situation where a project contains a large number of artifacts, tool support is needed to assist in locating the different artifacts. It is also important for traceability information to be accessible by different tools.

Challenge and its Solutions in Literature: Only two of the reviewed papers mentioned this challenge [14, 104]. The solutions proposed is that the company, through tools, should ensure that users have all the necessary information and proper access to the artifacts needed to create traceability links. Tools should provide features such as search by ID or search by keywords, to make it easy for the users to find the artifacts they need.

Challenge and its Solutions in the Automotive Domain: Only 7 of the multi-vocal sources report this challenge. The proposed solution is to collect all relevant development information in a centralized data storage.

Comparison to Case Company: For the case company, this challenge is partially solved as the tools used have the ability to search for and locate specific artifacts in an easy way. For traceability links involving artifacts stored in different tools the user still needs to copy the ID manually from one tool to another. While users have access to the artifacts needed due to the presence

of centralized storage with appropriate access rights, it is still not possible to access traceability information stored in the requirements management tool (in this case DOORS) directly from other tools.

2.5.1.4 Diverse Artifacts and Tools

Description: In the software development life cycle different tools are used for the different development activities such as requirements engineering, system design and so on. This means that artifacts are of different formats. Furthermore the artifacts specified in the different tools can contain redundant information which leads to inconsistencies when the system evolves as only some of the artifacts are updated. Development artifacts, especially requirements, can also be specified in various languages. Most traceability tools either do not support linking to artifacts located outside the tool or only support linking to specific tools or specific formats.

Challenge and its Solutions in Literature: Ten of the reviewed studies report this challenge [5, 8, 14, 22, 38, 80, 107, 108, 119, 134] From the studies, there are two different solutions for this challenge. The first option is to use an integrated tool platform that supports all the development activities. A user can interact with heterogeneous artifacts in such an environment using the same user interface and functionality. It includes traceability functionality and the ability to create traceability links between these heterogeneous artifacts. The second solution is tool integration where all existing tools are integrated so that it is possible to exchange information about the heterogeneous artifacts and create traceability links between them. This is however not a trivial task and requires a considerable effort, especially if there are many tools that need to be integrated [26].

Challenge and its Solutions in the Automotive Domain: This is the most reported challenge by the multi-vocal literature. It has been reported by 74 sources, where two of them report that in German automotive companies, some requirements are in English while others are written in German which further complicates traceability [112, 135]. Two of the solutions proposed are similar to what was proposed in the tertiary review. An additional solution is to define all the artifacts in a structured way so that they can be easily traced. This can be done for instance by specifying artifacts as formal models (e.g., [136]), tagging artifacts with traceable tags (e.g., [137]), by enforcing naming conventions (e.g., [138]), or by using an integrated modelling language. In this case, homogeneous artifacts are created in one specific modelling language. The model elements can be linked to each other through constructs of that modelling language. Several tools can interact with the artifacts (e.g., [139, 140].

Comparison to Case Company: In the case company, a total of eight tools are used for different development activities. Tool integration is a technically challenging task. Therefore, the company currently uses implicit links to link to artifacts in different tools which are created by copying IDs from one tool to another. This is not only time consuming, but also error prone and does not allow for any analysis to be done on the links. To overcome this problem, the company is planning to acquire an integrated tool platform that will be able to store all of their artifacts and thus make them accessible for creating

traceability links. The main drawback of this solution as reported by one of the architects is that it is hard to find a holistic tool that fully supports all the activities in the development life cycle. Currently, there are no holistic tools supporting activities like simulations which means that even with the holistic tool in place, other tools will still be used. Therefore this challenge is partially solved as linking to tools outside the holistic tool requires implementation of special plugins, which is costly in terms of time and might require rework as the involved tools evolve.

2.5.1.5 Manual Link Creation and Maintenance

Description: The task of creating traceability links is time consuming when it is done manually. This is exacerbated when there is a large number of artifacts involved. Moreover, traceability links become outdated when the artifacts they connect evolve. This means that they need to be updated in order to remain correct. Manually updating them is time consuming and error prone.

Challenge and its Solutions in Literature: This is one of the most frequently reported challenges in the tertiary literature review. It has been reported by 14 out of 24 papers [5, 8, 9, 14, 22, 34, 38, 61, 80, 108, 118, 125, 134, 141]. To overcome this challenge, the literature proposes the use of automated techniques to generate and update the traceability links. Examples of these techniques are: machine learning [21], information retrieval [9], event-based techniques [108] and model-driven techniques [134]. Most of the studies reporting these approaches have been on a theoretical level with small examples and using students as test subjects. For instance the literature review conducted by Borg et al. on information retrieval approaches for recovering traceability links shows that out of 34 publications studied, only one had an industrial evaluation [25]. Additionally, for automated techniques to work, implicit links have to be present so that the algorithms can use them to generate explicit links. In many cases, these implicit links do not exist due to lack of a uniform structure (e.g., naming schemes, meta-data) in the different artifacts.

Table 2.1: Search strings used in the tertiary and multi-vocal literature review

Data Source	Search String	Number of results
Tertiary Review		
IEEE Xplore	(‘‘Literature Review" OR Review OR Survey OR ‘‘Literature Survey") AND Traceability	160 results
SCOPUS	(Literature Review OR Review OR Literature Survey OR Survey) AND Traceability	40 results
ACM Guide	(Literature Review OR Review OR Literature Survey OR Survey) AND Traceability	322 results
Multi-vocal review, RQ 2a		
IEEE Xplore	(((((Automotive SPICE) OR ASPICE) OR ISO 26262) OR ISO26262) AND Traceability)	8 results
SCOPUS	((aspice OR automotive AND spice OR iso 26262 OR iso26262) AND (traceability))	25 results
ACM Guide	+(ASPICE "Automotive Spice" "ISO 26262" ISO26262) +(Traceability)	15 results
Google Web Search	(aspice OR "automotive spice" OR "iso 26262" OR iso26262) AND traceability	approx. 34700 results (263 after filtering)
Google filters the search results to exclude similar results, parantheses and AND operator added for clarity.		
Multi-vocal review, RQ 2b		
IEEE Xplore	((((Automotive Traceability) NOT rfid) NOT barcode) NOT laser) in the time span between 2008 and 2018	40 results
SCOPUS	(TITLE-ABS-KEY (automotive AND traceability) AND NOT TITLE-ABS-KEY (manufacturing) AND NOT TITLE-ABS-KEY (rfid)) AND (LIMIT-TO (SUBJAREA , "ENGI") OR LIMIT-TO (SUBJAREA , "COMP")) AND PUBYEAR > 2008 AND PUBYEAR < 2018	163 results
ACM Guide	+(Automobilindustrie Automotive Automobil) +(Traceability Nachverfolgbarkeit RÃijckverfolgbarkeit Verfolgbarkeit) -Manufacturing -Food -Laser -Barcode -"supply chain" -"logistics chain" -"lot tracking" -chargenrÃijckverfolgung -rfid) in the time span between 2008 and 2018	95 results
Google Web Search	((((Automobilindustrie) OR Automotive) OR Automobil) AND ((((Traceability) OR Nachverfolgbarkeit) OR RÃijckverfolgbarkeit) OR Verfolgbarkeit))	approx. 189000 results (356 after filtering)

Table 2.2: Challenges and solutions for traceability in the automotive domain

Challenge	TR	MLR	Found at Company	Challenge Solved?	Solutions
Knowledge of Traceability					
Lack of knowledge about and understanding of traceability	11	13	Yes	Yes	Training, Updated guidelines from certification bodies
Difficult to define information model	7	7	Yes	Partially	Defined traceability information model, Updated guidelines from certification bodies
Level of granularity	3	17	Yes	Yes	Defined traceability information model
Unclear traceability process	10	28	Yes	Partially	Defined traceability process, Defined traceability information model, Structured information, Integrated tool platform, Tool integration
Tools					
Lack of Configurable Tools	6	13	Yes	Yes	Flexible tools
Confidence in Tools	3	4	No		Certified Tool Suite
Inaccessibility of Artifacts	2	7	Yes	Partially	Centralized data storage, De-centralized data storage, Flexible tools
Diverse Artifacts and Tools	9	74	Yes	Partially	Integrated tool platform, Tool integration, Integrated modelling language, Structured information
Manual work	14	50	Yes	No	Automation, Just enough traceability, Integrated tool platform, Integrated modelling language
Human Factors					
Misuse of Traceability data	3	1	No		Training
Perceived as an overhead	5	15	Yes	No	Automation, Report generation tools, Just enough traceability
Organization and Process					
Distributed software development	2	11	Yes	Yes	Centralized data storage, De-centralized data storage
Traceability Across Lifecycle Phases	1	35	No		Integrated tool platform, Defined traceability process, Automation, Integrated modeling language
Reuse of Traceability Information	0	6	No		
Uses of Traceability					
Trace links are almost never consulted or used	4	9	Yes	Partially	Report generation tools, Just enough traceability
Lack of proper visualization tools	6	12	Yes	No	Report generation tools
Measurement of Traceability					
Assessing the traceability maintained	5	8	Yes	No	Automation, Defined traceability process, Defined traceability metamodel, Structured data
Return on Investment (ROI).	8	13	Yes	No	Cost-benefit models, Just enough traceability, Automation
Exchange of Traceability Information					
Lack of Coordination in traceability activities	3	23			Collaboration tools, Defined traceability process
Lack of interchange standards	4	8	Yes	No	Common standard
Conflicting objectives	1	1	No		Defined traceability process
Confidentiality Constraints	2	6	Yes	Partially	

Challenge and its Solutions in the Automotive Domain: This is also one of the most reported challenges in the multi-vocal literature (reported by 50 sources). Just like in the tertiary review, automation has been suggested as a solution for this. An additional constraint for using information retrieval techniques is that in many German automotive companies, the requirements are written both in English and in German which makes information retrieval difficult. Further solutions are having "just enough traceability" (e.g., [142,143]), meaning that only links that are needed should be created and maintained. Another solution is to use an integrated tool platform or an integrated modelling language. If all artifacts are accessible from the same tool, then the work of locating artifacts when creating links is reduced (e.g., [144]). An integrated tool also makes it easier to track changes.

Comparison to Case Company: Interestingly, none of these solutions was viable for the company. In general, machine learning, information retrieval and event-based techniques have a low precision and therefore the chance that false traceability links are generated is high. Given that the company produces safety-critical systems and the traceability links are also used for the certification process, false links are not tolerable. Model-driven techniques, on the other hand, require that all the artifacts being linked to and from are represented as models which is not the case for the company, where only some of the artifacts are models.

2.5.2 Human Factors

In this category we found two challenges that have been reported in the studied literature. Only one of these challenges was found at the case company.

2.5.2.1 Misuse of Traceability Data

Description: This challenge refers to the fact that in some situations, people responsible for creating and maintaining the traceability links have a fear that this data may be used against them, e.g., during performance appraisals. This happens especially when developers need to create links from artifacts they are responsible for, e.g., bugs reported by users.

Challenge and its Solutions in Literature: This challenge has been reported by three of our reviewed literature [8, 38, 107]. The authors describe that employees have a fear that traceability data can be used against them and threaten their job security. This is an inappropriate use of traceability data as the data is supposed to be used for quality assurance of the system rather than used for judging employees' performance. The studies propose that both management and employees need to be educated on what traceability is and what the potential benefits are.

Challenge and its Solutions in the Automotive Domain: Only one of the sources in the multi-vocal review reported this challenge. According to [145], engineers fear that if they document everything they might become redundant and become replaceable. However, this source does not report any solution to the challenge.

Comparison to Case Company: At the case company, this was not part of the challenges that we identified. However, the company has a system that

already logs user activities with respect to creating and modifying development artifacts. If there is a problem in the system it is easy to identify who was working on the artifact and contact them about the problem. This data is not used for performance appraisals. This indicates that the development environment is already very transparent thus employees do not fear the misuse of traceability links.

2.5.2.2 Perceived as an Overhead

Description: In situations where traceability links are created manually, developers usually perceive this as an extra activity that they need to do or view it as a task that interrupts their workflow. Furthermore, this is a problem since the creators of the links are often not the ones using them. Developers therefore become demotivated and assign a low priority to this task, which can lead to either wrong or missing links.

Challenge and its Solutions in Literature: Five of our reviewed studies report this challenge [14, 34, 107, 108, 123]. Proposed solutions for this problem are to ensure that the traceability links created provide immediate benefit to the creators and also to automate the tasks whenever possible. This can be done with tools that enable quick navigation from one artifact to another or visualization techniques that give users an overview of the connection between different artifacts.

Challenge and its Solutions in the Automotive Domain: This challenge was reported by 15 of the sources in the multi-vocal literature. The main problem is that the creators of links are not the ones benefiting and therefore find the task demotivating (e.g., [122, 146, 147]). Suggested solutions are similar to those proposed in the tertiary review.

Comparison to Case Company: At the case company this is a challenge, due to the diversity of tools and the fact that implicit links are created between artifacts in different tools. It is hard for developers to get an overview of the traces. Across tools they still have to find artifacts by searching for ID and thus do not see the immediate benefits of traceability. All of the interviewees pointed out that being able to navigate easily using the traceability links and having graphical representations of how everything is connected would be a feature that would encourage them to create more correct and complete traceability links. Allowing for easy navigation across tools requires integrating the tools which is also not a trivial task as previously discussed.

2.5.3 Organization and Processes

In this category, we found three challenges, two of which have been solved at the case company and one which was not reported at the case company.

2.5.3.1 Complexity Added by Distributed Software Development

Description: In large organizations, it is a common phenomenon that development activities are carried out at multiple sites. This adds complexity to traceability, especially when the different sites need to share the development artifacts. Unless the infrastructure is set up correctly and the sites have a unified software development process, it can be very hard to create traceability

links. For companies distributed in various countries, different time zones and languages used in the different locations also make traceability establishment difficult.

Challenge and its Solutions in Literature: This challenges has been reported by two of the reviewed papers. These papers propose a centralized repository for storage of all development artifacts [8,14]. This way, the location of the developers will not matter as everything is centrally stored and shared. Such a repository also needs to be guarded by an access control system to make sure that the right people have access to the artifacts they need.

Challenge and its Solutions in the Automotive Domain: Eleven of the multi-vocal sources report this challenge. The solution proposed is again to use a centralized data storage where all artifacts are stored and therefore accessible by the staff in different locations (e.g., [148, 149]). Another solution is for the company to put in place means of communication and collaboration between the teams in the distributed locations [142]. This can be done by using tools that provide collaborative features such as chats and comments on artifact level.

Comparison to Case Company: The company has solved this challenge by having centralized repositories where the artifacts can be stored and different developers are given access rights accordingly. This is in line with what the scientific literature proposes.

2.5.3.2 Traceability Across Lifecycle Phases

Description: Traceability needs to be established between artifacts that are produced at different stages in the development lifecycle. In principle this is defined in the traceability process (cf. 2.4.1.4). However, even if such a process is in place, there is still a gap between these lifecycle phases, mainly because they are performed in isolation with different teams and people. It is also common that there is no direct mapping between the artifacts produced in the different phases.

Challenge and its Solutions in Literature: Only one of the sources in the tertiary review reports this challenge [134]. One solution has been suggested which is to have a defined traceability process that is supported by tools, e.g., an integrated modeling language that defines which links should be established between models in the different lifecycle phases.

Challenge and its Solutions in the Automotive Domain: This challenge was reported by 35 of the sources in the multi-vocal literature. Several solutions have been reported that can contribute to solving this challenge of both the development process and tools. First of all having an integrated tool platform or an integrated modeling language that integrates all the phases in the development lifecycle ensures that the different artifacts from the different phases are accessible (e.g., [150, 151]). When an integrated tool platform is not possible, integration of the different tools is suggested via technologies like OSLC to ensure that the different phases are connected tool-wise (e,g., [152]). Furthermore a well-defined traceability process and a traceability information model should be put in place and enforced by the development companies (e.g., [127, 153]). People performing the different activities in the different phases should be aware of their roles and responsibilities when it comes to

traceability. Lastly automation can help solve this challenge by, e.g., using model-driven techniques to generate artifacts or skeletons of artifacts from one development phase to the next (e.g., [154, 155]).

Comparison to Case Company: In the company, this challenge has been solved. Even though the interviewees reported that there is diversity in the tools used in the different lifecycle phases, the development process is well defined and enforced in the company. For instance code will only be written if there is a low-level (detailed) requirement associated with it. This means that the different phases are connected and hence this is not a challenge.

2.5.3.3 Reuse of Traceability Information

Description: It has already been discussed that establishing traceability is a manual and time consuming process. It is therefore an advantage if the established links can be reused in similar projects or when parts of the projects are being reused, especially in product line environments. This is currently a challenge as it is not clear how to select relevant information for reuse without introducing links outside of the reuse scope. If, e.g., an architectural component should be reused, selecting which of the traceability links connected to it (and thus, which other artifacts) should also be reused is currently a task that is not supported by tools or guidelines.

Challenge and its Solutions in Literature: None of the sources in the tertiary review reports this as a challenge.

Challenge and its Solutions in the Automotive Domain: This has been reported by six of the sources in the multi-vocal review. In the automotive domain, in most cases, systems are not build from scratch but rather reuse existing artifacts such as requirements and code. Developers and stakeholders therefore would like to make use of traceability information when reusing artifacts. Unfortunately, none of the sources reports solutions or best practices. This shows that this is a topic that needs further research.

Comparison to Case Company: At the company, this challenge was not reported by any of the interviewees and also not observed in the process. Currently, traceability information is not reused.

2.6 Results: Outcome

In this section, we report on challenges related to the outcome of the traceability process. The section is divided into two subsections which are *Use of Traceability* containing challenges encountered when using traceability links and *Measurement of Traceability* containing challenges associated with measuring the quality and benefit of the traceability links.

2.6.1 Uses of Traceability

For this category, we found two challenges. One of the challenges has been partially solved and one challenge is unsolved.

2.6.1.1 Traceability Links are Almost Never Used

Description: Even with the large amount of time and effort invested in establishing traceability, traceability links are not used at all or under-utilized. The main use of traceability is still for certification.This is mainly due to the following: 1) lack of tools that facilitate utilization (for instance, good visualizations); 2) the number of links is too high and therefore unusable; and 3) lack of trust in the quality of the traceability links.

Challenge and its Solutions in Literature: This challenge has been reported by four of the reviewed papers [9, 34, 108, 123]. In [108], it is reported that traceability links are not used either because the links recorded are not helpful to support development activities or because the tools do not provide an efficient way of using the links. The authors point out the importance of tailoring traceability according to the needs of the users and not just creating traceability links for every artifact. In [123], the authors point out common flaws that cause traceability links to be ignored. These flaws are, e.g., redundant traceability paths, missing links and out-dated links.

Challenge and its Solutions in the Automotive Domain: This challenge was reported by nine of the sources in the multi-vocal review. Again the most common use of traceability is for certification purposes. The solution proposed in the multi-vocal literature are similar to the ones proposed in the tertiary review.

Comparison to the Case Company: At the company, the main driver for establishing traceability is the requirement from OEMs to be ASPICE compatible. Therefore the main use of the traceability links is for certification purposes. During the interviews we also found that traceability links are used to track the progress of the project, for instance, to check how many requirements already have test cases. The architects and developers however noted that they would like to utilize the links more but that there is no convenient way to do that at the moment. For instance, it is sometimes necessary to copy IDs from one tool to another to search for the connected artifact. This makes it very hard to get an overview of the system or feature through the traceability links. This challenge is therefore partially solved and would be fully solved if better tools that facilitate usage of traceability links are put in place.

2.6.1.2 Lack of Proper Visualization and Reporting Tools

Description: When traceability is properly established, it can result in a large number of links, in particular if the project consists of a large number of artifacts. The end users of these links need proper visualization tools in order to understand them and powerful reporting tools to produce overviews and statistics for reviews. This is currently a challenge as traceability links are usually presented in large tables or lists where it is hard to comprehend what they mean and even harder to detect flaws in them.

Challenge and its Solutions in Literature: This challenge was reported by six of the reviewed papers [61, 80, 110, 118, 123, 156]. In [80], the authors point out that it is very important, especially with automatically generated traceability links, to have meaningful graphical representations so that traceability links can be easily inspected for inconsistent and outdated links. Visualization techniques that will facilitate development activities are

proposed in [61]. For instance, it is useful to have a visualization that will allow the user to see which requirements are already implemented and tested or which tests do not have corresponding requirements.

Most common visualizations of traceability links are a matrix, graphical notations, and hyperlinks. In the matrix view artifacts are displayed in a table with a mark on the cell where the artifact in the column and that in the row are connected by a traceability link. The graphical view represents the artifacts as nodes and the links as edges in a graph. In the hyperlinks view, traceability links are displayed as hyperlinks from an artifact and can be clicked to navigate to the linked artifacts. The authors in [108] propose that a traceability tool should have a combination of the three representation as all have advantages and disadvantages and are used for different purposes. The authors illustrate that a project manager may only need an overview of the project but a developer making a change to the system may find hyperlinks more useful as navigation to and from artifacts is facilitated [108].

Challenge and its Solutions in the Automotive Domain: This challenge has been reported by twelve of the sources in the multi-vocal review. The solution suggested is also similar to the one suggested in the tertiary review which is, tool vendors need to develop tools that allow custom reports to be generated from traceability information based on user needs (e.g., [157–159]).

Comparison to Case Company: In the case company, this was also reported as a challenge that is not solved. This was mainly noted by the developer and the architects who suggested that the traceability links would be more useful for them if they had better graphical representation. They specifically asked for visualization where one is able to get an overview of the project or a specific feature through the traceability links. Also the traceability links that are created manually, for example by copying an ID of one artifact and adding it in another, are not supported by the visualization available in the requirements management tool used in the company.

2.6.2 Measurement of Traceability

For this category, we found two challenges, both of them unsolved.

2.6.2.1 Difficult to Assess the Quality of Traceability Links

Description: When traceability is properly established, it can result in a large number of links. In order to trust and use the traceability links, it must be possible to assess their quality by, for instance, measuring how correct and complete the set of traceability links is. This is a challenge as the most reliable assessment method is still manual checking.

Challenge and its Solutions in Literature: Five of our reviewed papers note this as a challenge [5, 34, 61, 119, 156] It is hard to assess if the traceability maintained is of high quality as reported in [123], where the authors note that even in safety-critical domains the traceability links submitted for certification contain either missing links or redundant links. In [5], it is reported that especially for generated traceability links, it is a challenge to evaluate their correctness and completeness. One proposed solution is to attach confidence values to the generated link and have a threshold based on the confidence

value to determine which links are correct. However, this approach does not guarantee that the links will be complete or correct. Another solution is to use the semantics defined in the traceability metamodel to assess the traceability links. For instance, if the information model defines that every requirement should be linked to a test, then missing links can be detected by checking if all requirements have a link to a test. This however only guarantees finding missing links, completeness and correctness still needs to be checked manually.

Challenge and its Solutions in the Automotive Domain: Eight of the multi-vocal sources report this as a challenge. In the automotive domain, it is unclear to companies how traceability links can be assessed to ensure compliance with safety standards. This is because of the lack of guidelines on assessment and in some cases inconsistent guidelines and conflicting requirements from different standards [116]. Several solutions are suggested to tackle this challenge. One of them is using semantics of the defined traceability information model as suggested by the tertiary review. Additionally, the multi-vocal review suggests having structured data that can be checked (e.g., [116, 160]). For instance if high-level requirements and low-level requirements have unique naming schemes, then it can be checked that a high level requirement is indeed linking to a low level requirement. Another solution is to define the assessment strategy of trace links when defining the traceability process (e.g., [116, 161]). Even if the strategy is a manual one (e.g., reviews by developers), if it is well-defined and enforced it can improve the quality of the links.

Comparison at the Case Company: At the case company, this is currently one of the unsolved challenges. For traceability links that are created between artifacts in DOORS, there is a possibility to check for missing links easily since the tool allows identifying requirements with no links. Also, since the tool supports defining custom trace links, it is possible to limit which kinds of artifacts a link can connect. The advantage of this feature is that it prevents the creation of links that are semantically wrong. For links that are created with artifacts that are not in DOORS this kind of check is harder as it requires implementation of extra plugins that can do such checks. Correctness and completeness on the other hand needs to be checked manually. This can be done during review meetings but consumes a lot of time and effort.

2.6.2.2 Difficult to Measure the Return on Investment

Description: Since the most common way of establishing and maintaining traceability in practice is manually, this is a cost-intensive task that requires the company's investment both in terms of money for the tools and in terms of time. It is therefore important for a company to be able to measure what the return on investment of the established traceability links is. This is a challenge as the cost is significant while the benefits cannot be easily measured.

Challenge and its Solutions in Literature: Eight out of the reviewed papers report that traceability establishment is an expensive process [5, 8, 9, 14, 22, 38, 108, 118] This is because developers need to spend extra time to create and maintain traceability links. Most managers think that a project that implements traceability is more expensive than one which does not [38]. Currently there are no measurements that can provide evidence of these direct benefits of traceability. Research proposes cost-benefit models that can be used

to show how much traceability has contributed to activities such as maintenance and understandability [85], but these still need to be validated in practice. This is not a trivial task as such benefits are mostly visible at the end of the project. To minimize the effort spent on traceability creation and maintenance, researchers have proposed having "just enough traceability" where links are created only to artifacts of high value (e.g., high priority requirements) [8].

Challenge and its Solutions in the Automotive Domain: This challenge has been reported by 13 sources in the multi-vocal review. Automotive companies have difficulties proving that traceability is beneficial especially for cases where full time employees are dedicated to this task [162]. The multi-vocal literature does not suggest any cost benefit models but rather suggests that there is a need to have more cost-effective ways of establishing traceability. This can be through automation of traceability tasks where possible and also by creating just enough traceability, only links that are needed should be created and maintained [142].

Comparison to Case Company: The results of the case study indicated that this challenge has not been solved. All of the interviewees including the managers confirmed that they think traceability is expensive and they do not have evidence of the value it adds to the projects. The only reason that justifies investing in traceability is because it is a mandated task, they have to do it. Value-Based Traceability is also not a feasible solution for them as *full* traceability is a mandatory requirement for safety-critical applications. It is also hard to maintain an exclusive list of high priority requirements that need traceability as priorities can rapidly change over time.

2.7 Results: Exchange of Traceability Information

This section reports challenges associated with how traceability information can be interchanged between teams within an organization and between different organization.

2.7.1 Exchange Within and Across Organizations

In this category we found four challenges from the literature. At the case company, one challenge is partially solved even though there was no proposed solution in literature , one is not solved and two of the challenges were not observed.

2.7.1.1 Lack of Interchange Standards

Description: To facilitate the sharing and transfer of traceability information from one company to another, there is a need for a common standard. Currently, such a standard does not exist and traceability information is stored in various forms ranging from implicit links established through copying IDs from one artifact to another, to explicit traceability links that utilize formal notations such as models. Some links are also stored together with the artifacts while others are stored in a separate trace model with only references to the connected

artifacts. Depending on the tool the formats of the traceability links can also vary substantially. This makes it difficult for traceability to be exchanged and reused in different companies.

Challenge and its Solutions in Literature: Four sources in the tertiary review report this challenge [31, 34, 119, 156]. The literature proposes the need for one standard that can be used by companies in order to facilitate this sharing and exchange of traceability information [34].

Challenge and its Solutions in the Automotive Domain: This challenge was reported by eight of the sources in the multi-vocal review. The OEM and supplier relationship in the automotive domain means that artifacts are exchanged between the two companies. Some of these artifacts contain traceability information. If there is no standardized format for the links, then they are inaccessible. It is reported in [163] that OEMs sometimes acquire entire subsystems from suppliers but have no way of accessing the traceability information from these subsystems. The solution proposed here is similar to the one proposed by the tertiary review. A common standard of accessing and exchanging traceability information is needed. The multi-vocal review suggests that OSLC can be a common standard for information access, but the question of common semantics is still open.

Comparison to Case Company: This is a challenge that the company faces. For instance, OEMs can send requirements which could have traceability links as well. But if the tools at the company cannot identify these links then that information is lost and has to be re-created from scratch.

2.7.1.2 Conflicting Objectives

Description: When more than one company is involved in the development of a system, it is important to align organizational objectives of all the companies. This is true also for traceability. If the objectives for traceability in one company contradict the ones in another, there might be a conflict. For instance, if the supplier and OEM created traceability links that are not compatible (in terms of types and granularity), then the links end up being unusable between the organizations because each organization has a different objective. If the OEM has the objective of using the traceability links from its suppliers in an aggregated manner to get an overview of the entire system, this will only work if all of the suppliers create the needed traceability links.

Challenge and its Solutions in Literature: Only one of the reviewed papers [31] reports this challenge. It proposes that at the beginning of the project, all the stakeholders need to align their objectives, including traceability objectives. It is important to define early on what each stakeholder requires and is expected to deliver in terms of traceability.

Challenge and its Solutions in the Automotive Domain: In the multi-vocal review only one source reports this challenge. The OEM and the supplier may have different objectives that can be conflicting [142]. The solution proposed is similar to the one proposed in the tertiary literature review.

Comparison to Case Company: This challenge did not come up in the study at the company. Since the company is a supplier, one of their objectives is to satisfy the OEM. In this case, the demand for traceability actually comes from the OEMs. The OEMs specifically asks the company to be compliant to

the ASPICE standard in which traceability is one of the requirements.

2.7.1.3 Confidentiality Constraints

Description: Establishing traceability links that cross the organizational boundaries is a challenging task due to confidentiality implications. It is difficult for suppliers for example, to create traceability links when some artifacts are not accessible to them since they are confidential due to protected intellectual property from the OEM.

Challenge and its Solutions in Literature: In the reviewed literature, two of the papers [31, 104] mention this challenge but there are no proposals for how to establish traceability when the artifacts are restricted due to legal reasons.

Challenge and its Solutions in the Automotive Domain: This challenge was reported by 6 sources in the multi-vocal review. Most of the time the suppliers only receive partial requirements which makes traceability harder [164]. Only one source suggests a solution [165] where the development process, including all artifacts exchanged between the OEM and supplier, should be transparent. This can be hard to implement since OEMs keep some artifacts confidential, e.g., because they contain intellectual property that distinguishes them in the market.

Comparison to Case Company: The company also faces this challenge when some of the artifacts they want to trace to cannot be shared by the OEMs. Currently they do not have a solution for this. For some OEMs, the company shares requirements via web interfaces. The OEMs can then limit which fields are visible to the OEM and which fields are visible to both the supplier and OEM. This is an initiative towards sharing confidential information.

2.7.1.4 Lack of Coordination in traceability activities

Description: During software development different roles need to coordinate. This becomes more important in system development because various parts of the system are developed by different disciplines, from different companies and have to be integrated in the end. For example the software team needs to coordinate with the hardware team to make sure that their software will work on the hardware. This coordination is also important when it comes to updating the traceability links. Coordination becomes difficult because the different disciplines use different vocabularies, have different objectives, and most of the time the development is isolated. When development is done across companies, the different companies involved may also have different development processes.

Challenge and its Solutions in Literature: This challenge was observed by four of the papers we reviewed [8, 9, 31, 104]. In [9], just enough traceability (value-based traceability) is proposed as a means to reduce the amount of links created and hence reduce the time people need to coordinate on traceability link maintenance. In [104], the authors report that change notification is useful for coordination. When an artifact connected by a traceability link has changed, the person responsible for the it should be notified in order to decide how the link should evolve.

Challenge and its Solutions in the Automotive Domain: 23 of the sources in the multi-vocal review report this challenge. Systems development involves various disciplines and establishing traceability between artifacts from the different disciplines is difficult if the disciplines do not collaborate. In the automotive domain this challenge becomes more complex due to the OEM and supplier relation where parts of the tracing need to be done at the OEM and parts need to be done by the supplier [112]. There is currently no defined process on how to do this. Two solutions have been suggested. One is to have tools that support the different disciplines with collaboration features such as chats, forums and notifications. This can be part of an integrated tool platform. Second is having a defined process on how the teams should collaborate, in [SG62], it has been suggested that cross-discipline work assignments should be designed to make the different disciplines collaborate more.

Comparison to Case Company: At the company this was not observed as a challenge. On further analysis this can be due to the fact that the requirements management tool has a feature called "suspect links". It highlights the links that connect artifacts which have changed. The user can thus investigate the change and decide how to update the traceability link and the connected artifacts. When working as a team, the suspect links are also propagated to a developers local workspace when they pull changes from the repository. The developers can navigate to see what has changed in connected artifacts by clicking the suspect links.

2.8 Discussion

In this section, we discuss our results in relation to the research questions. We will address RQ 1 and RQ 2 that deal with the general traceability challenges and the particular challenges of traceability in the automotive domain in sections 2.8.1 and 2.8.2. RQ 3 that addresses challenges that can be observed in practice will be discussed in Section 2.8.3.

2.8.1 Differences between the tertiary and the multi-vocal review

While most of the challenges and solutions found in the tertiary review were also found and thus confirmed in the MLR, there are a few differences that stand out. This has partially to do with the different data sources and the different provenance of the information (discussed in Section 2.8.2) and partially with the fact that the sources in the MLR were more specific to the automotive domain. One challenge has been newly identified from the MLR sources: *Reuse of traceability information*. In addition, *Traceability across lifecycle phases* has only been reported once in the tertiary review, but 34 times in the MLR sources.

The reason why *Reuse of traceability information* was not identified as a challenge from the secondary literature might be due to the focus of the MLR on the automotive domain and the high maturity of product line approaches in this area [166]. Reuse of traceability information is usually described in this context. When a component or a subsystem has to be reused in another

product, all attached requirements, design documents, test cases, etc. should also be accessible to the developers of the new products. Since these artifacts are connected via traceability links, this information must also be reused. However, there might be traceability links present to the artifacts of other components that should not be reused. This introduces a challenge in terms of which links to reuse and how to deal with those links that point to targets outside of the reused artifacts. It is not clear which solutions apply to this challenge at this point.

The challenge of *Traceability across lifecycle phases* is one of the most reported challenges in the MLR (cf. Figure 2.5). Our analysis shows that out of the 34 sources that report this challenge, 16 are written by tool vendors. This might be due to the fact that one of the selling points for tools is the ability to establish traceability across all development phases (cf. Section 2.2). This also correlates with the fact that the sources that mention this challenge also claim to provide a solution either in terms of an *integrated tool platform* or *tool integration*. However, this seems to solve only one side of the challenge which is how the different tools in the different phases can work together. The other side of this challenge is the process side, which refers to how the people involved in the different phases should create and maintain traceability links. A solution suggested is to have a well defined traceability process even though the specifics of what this process looks like and how it should be established have not discussed.

With regards to solutions, there is again a significant overlap between the solutions proposed in the tertiary review and those in the MLR. However, two thing stand out: 1) While the *Confidence in tool* challenge has no concrete solution from the tertiary review, the MLR proposes to have a *certified tool suite* which has been cleared by a certification authority for use within a process to develop safety-critical systems. Again we think that this solution is used as a marketing point, given that it has been reported by only tool vendors and consultants; 2) On the one hand, the MLR calls for *updated guidelines from certification bodies* so that practitioners can have a clear understanding on what they need to do (in terms of traceability) to be able to comply with the standards. On the other hand, the tertiary review reveals that there is a need for guidelines and best practices from research on how to efficiently establish traceability. This shows that from both the academic and the practitioner side, the task of establishing traceability is still not well understood and requires the collaboration of both practitioners and academia to establish guidelines and best practices.

2.8.2 Differences by challenge and solution provenance

The provenance of the different challenges and solutions proposed in the regarded sources refers to which stakeholder the source can be traced to. There are marked differences between issues discussed in scientific literature that mostly stems from academics and from teams that are a mix between academics and practitioners and the reports made by tool vendors, consultants, and users of traceability. Unfortunately, the latter category is not well represented in this study with only 27 of 246 sources directly attributable to users. Interestingly, of these 27, 15 are publications in peer-reviewed venues.

Figure 2.5: Distribution of challenges by provenance in absolute numbers.

This indicates that information about state of the practice from the user perspective is available in the scientific literature.

Tool vendors want to push the features of their tools and provide mostly marketing material online. However, they are responding to the needs of their customers, so that the features that traceability tools provide reflect (at least in part) issues that the industry deals with. This can be seen, e.g., in the focus on *report generation* and integration, where the latter is addressed with either an *integrated tool platform* or *tool integration*. Features for *flexible visualisation and report generation* are a response to the challenge that is posed by a lack of such tools. Likewise, the two integration approaches are a response to the challenge of *diverse artifacts and tools*. The fact that these challenges and solutions are reported by sources of all provenances indicates that there is an agreement about the validity of the challenges and the potential solution approaches to them.

The challenges *Misuse of traceability data* and *Conflicting objectives* showed up only once, both times in academic papers. Since these are not reported as challenges by any of the practitioners or are even accepted as widespread in the scientific community indicates that they might not be general problems but have rather been observed on few occasions and might thus be issues of individual companies or even project teams. In terms of *Conflicting objectives*, the automotive industry might also be a special case: OEMs and suppliers in most cases have very long-standing relationships with clear communication channels. It can be expected that the objectives are fairly aligned in such an environment. In addition, ASPICE is indeed a standard to regulate the relationship with the supplier. If a supplier follows the standard, any conflicts between expectations and what is delivered should be minimized.

2.8.3 Unsolved Challenges at the Case Company

With regards to RQ3 (*Which of the reported traceability challenges in scientific literature and non-scientific literature can be observed in practice in the automotive domain and how have they been solved?*), the findings reported in Section **??** show that there is a total of six unsolved challenges. An overview of this is given in Figure 2.4. We will focus our discussion on the unsolved challenges and why they are so difficult to address since this sheds light on how the special circumstances in the automotive domain influence the applicability of solutions. Table 2.3 gives a summary of the persistent challenges at the case company, why the solutions from the literature are not applicable and which extensions we propose to solve the issues.

One partially solved challenge, however, deserves some attention: *Diverse artifacts and tools* was the most reported challenge in the MLR. The company has integrated tools where possible so that links can be created to and from artifacts in different tools. For instance, in one team, the requirement tool (DOORS) has been integrated with the design tool (Enterprise Architect). However, this is done only for some tools, to allow traceability to tools that have not been integrated, e.g., the requirements tool and the testing tool, the company has a structured way of naming artifacts uniquely, these unique names are then copied from one tool to another to create traceability links. Even though this is a manual process, it works because there are guidelines in how

these naming conventions work and the developers follow these guidelines.

Manual work (*Tools*): Several studies have focused on machine learning [21, 167], information retrieval [20] and rule-based techniques [23] for automating the creation and maintenance of traceability links. However, due to the fact that automated techniques can generate incorrect links, which is in violation to safety standards such as ISO 26262, they have not been adopted in the automotive domain. Furthermore, automation techniques only work if implicit links are already in place. To overcome the problem of incorrect links, researchers have proposed that generated links are manually inspected by humans. However, it has been shown that giving a set of generated links to humans to sort out incorrect links can even decrease quality [36].

Other automation techniques in literature are model-based techniques where traceability links are generated as a by-product of transformations. Model-driven traceability works if all artifacts are models. This is not necessarily the case in the automotive industry. Even if models exist, they are often independent and not connected by transformations. Second, many transformation tools that support the generation of traceability links have their own pre-defined notion of link structure and semantics. This makes it hard to integrate them in traceability tools already used in companies [22].

To practically solve this challenge, traceability tools have to enable the combination of manual, semi-automatic and automatic techniques for creation and maintenance of traceability links. Since each of these approaches has its advantages and disadvantages, they can complement each other. For instance, to make sure the links are correct one can rely on manual creation, but to reduce the effort of maintenance, automatic and semi-automatic techniques can be used. Semi-automatic techniques include sending notifications and warnings to users on traceability issues and suggesting probable solutions on how to fix issues. This kind of solution has been investigated in [29, 168] and the authors show that the solution is promising when properly integrated into the traceability tools.

Lack of interchange standards (*Exchange of Traceability Information*): For requirements, there already is a Requirements Interchange Format (ReqIF)[6], which is being adopted and provided as exports from several requirements management tools. Extending such a standard or creating a similar standard for traceability exchange can resolve this challenge. Several sources from the multi-vocal review suggest OSLC[7] as an interchange standard for traceability [117, 169,170]. OSLC is an integration technology which enables tools to integrate on the data level, i.e., data from one tool can be made accessible to another tool. With a proper set-up, tools from the OEM and supplier can make artifacts available via OSLC and hence enable creation and use of links across companies. It should be noted that a common standard will not solve the *Diverse artifacts and tools* challenge as data still needs to be shared between companies which can cause inconsistencies as the data evolves. Where not legally constrained, we encourage suppliers and OEMs to share the data repository to avoid such inconsistencies.

[6] `http://www.omg.org/spec/ReqIF/1.1/`

[7] Open Services for Lifecycle Collaboration, `https://open-services.net`

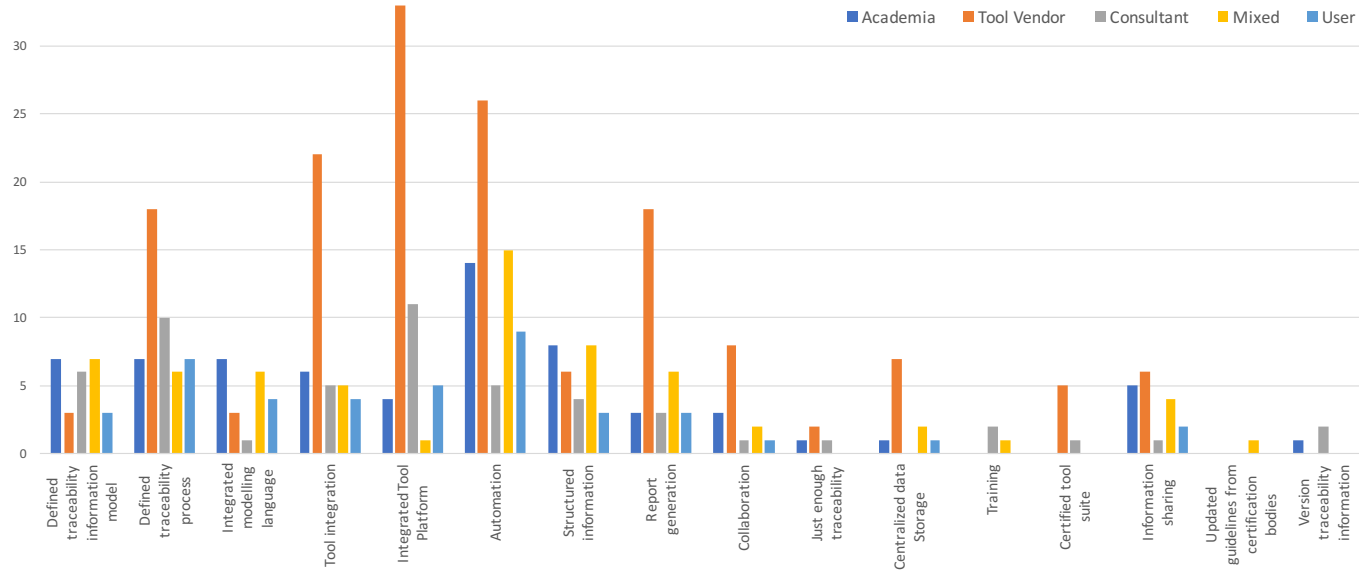


Figure 2.6: Distribution of solutions by provenance in absolute numbers.

Table 2.3: Challenges and proposed solutions.

Challenge	Solutions in Literature	Why solutions are not applicable	Proposed Extensions
Manual Work	Machine Learning [21] [171], Information Retrieval [25] [122, 172], Rule-based [23] and Model-based techniques [134] [173]	Machine learning and information retrieval techniques produce incorrect links; not acceptable for safety-critical systems due to the ISO 26262 standard.	Use semi-automatic approaches for maintenance (e.g., to push notifications of artifact changes to responsible users and suggest how links should be updated). Combine manual links with model-based techniques to create links.
Perceived as an overhead	Develop tools that require less effort and produce immediate benefits (e.g, ease of navigation), training on importance of traceability [14, 34, 107] [174] .	Large number of heterogeneous artifacts that need to be traced to. Traceability is viewed as only important for certification.	Complement the traceability process with gamification features. Developers can, e.g., be rewarded based on the number of correct links they create and projects can be awarded points/badges based on completeness of traceability links.
Lack of visualization tools	Matrix view, Graphical view and Hyperlinks [108] [159, 175]	Too many traceability links due to large and complex systems.	Provide visualizations suitable for end user needs. Develop tools that enable visualizations to be customised. Users should be able to create different views (graphs, charts, matrices, etc.) based on different data from traceability links.
Assessment of traceability	Use a well-defined traceability information model to facilitate checking for missing links and prevent invalid link creation [123] [162], event-based maintenance [168], text-matching	Distributed and isolated development phases	Extend event-based techniques to enable notifications to be sent to artifact owners when links are created involving these artifacts in order to facilitate correctness checks.
Return on investment	Value-based or "just enough" traceability [176] [142, 161]	Links have to be created from all safety-related requirements regardless of their value or priority	Monitor activities supported by traceability to automatically collect evidence on advantages of traceability. Communicate this evidence in the company.
Lack of interchange standards	Create a common traceability standard [85]	A common traceability exchange standard accepted by OEMs and suppliers does not exist.	Adopt OSLC as a way to access data in other tools. Define a common information model.

Lack of Visualization and Reporting Tools (*Use of Traceability*): At the case company, all interviewees were not satisfied with the visualization provided by their traceability tool. Our analysis shows that this is attributed to the fact that most tools are not well adapted to the requirements of using links in different scenarios. Instead, much of the effort in developing these tools is dedicated to the functionality of creation and maintenance of the links, rather than visualization. To solve this problem, we propose that there is a need to first analyze different use cases in which traceability links are used. A study by Bouillon et al. [4] investigated different scenarios in which traceability links are used. Conducting such a study in the automotive domain will lead to usage scenarios that can be used to determine which kind of visualization is appropriate for each use case. When this is clear, it will be possible to add such visualizations to existing tools and support the users when using traceability links. Additionally, tools should provide possibilities for users to create customized reports based on their needs [157–159].

Assessment of Traceability (*Measurement of Traceability*): This challenge refers to how the quality of the maintained traceability links can be measured to ensure that the links are both correct and complete. Measuring completeness is tricky since it is difficult to define what completeness means. For instance, while it is possible to check with tools that every requirement has a link to a test, it is not possible to determine that the tests actually cover all aspects of the requirement. Tools are able to flag requirements with no links to test cases, but it is still up to the developers to determine whether the linked tests provide sufficient coverage for the requirement.

Correctness is also hard to assess with tools. For instance a requirement can indeed be linked to a test but the decision if the test is a correct test for the requirement must be made in a time-consuming, manual process. Text-matching [177] that yields a similarity score between the connected artifacts is one approach that can be used to reduce the time spent on this task. The links below a certain similarity score threshold can be shown to the user for a manual check. This solution requires naming standards that ensure that there is always a text similarity between two connected artifacts. Another solution approach is to notify the owners of the linked artifacts when links are created. They can then raise their concern if they think the link is incorrect and discuss the link with the user who created it. This approach is similar to event-based traceability proposed in [178] where the authors suggest notifications to be sent to the owners of connected artifacts when one connected artifact evolves in order to update their artifacts, too.

Return on investment (*Measurement of Traceability*): Most literature on traceability points out benefits such as saving time and effort during impact analysis, tracking progress and improving understandability of the system. However, measuring these benefits in an industrial setting is not trivial because it is hard to isolate the effects of traceability. Also, some traceability benefits manifest only if the project has been progressing and is affected by, e.g., personel churn. Traceability can then save time by helping new developers understand the system and easily navigate to artifacts. Value-based traceability is one solution proposed to reduce the cost of creating traceability links [176, 179]. This means that when planning for traceability, companies need to assure that the links are useful for the project and thus beneficial by assessing why the

links are needed and how the benefit will be derived.

In the automotive domain, the main reason for adopting traceability is due to safety standards that demand traceability. This is however not a good motivation as traceability is adopted because people are forced to do it. Being able to quantify the benefits of traceability is one way to show that traceability is indeed useful. For this, we propose monitoring the activities that are supported by traceability links in the company in order to collect data on how useful traceability links are. Additional data can be obtained by conducting surveys with users of traceability and publicizing the results internally in order to promote its adoption in the company even for projects that are not safety-critical and thus controlled by safety standards.

Perceived as an overhead (*Human Factors*): This challenge has two aspects: an organizational and a technical one. The organizational issue is that the people creating and maintaining the traceability links are not the ones using them. A relation to the challenge of understanding traceability thus exists and sufficient training as well as the realization of the immediate benefits of traceability links can help in this regard. The technical aspect is related to the tools that offer little support in terms of visualization, navigation, and analysis. If, based on traceability links, the tools used in the industry can offer features such as easy navigation, visualization, customized reports or even recommendations for artifacts that can be re-used, then the developers creating the links will see their benefits. It should be possible to customize the tools in a way that benefits the creators of the links as well [86]. Another option is complimenting traceability tools with aspects of gamification to make the task of creating and maintaining the traceability links more motivating and engaging. This has been shown to work with other software engineering tasks such as requirements analysis and testing [81].

2.9 Threats to Validity

In this section we discuss the threats to validity of our study and ways in which we minimized these threats. We use the categories described in [64] but do not discuss internal validity as our study was not not examining a causal relation.

2.9.1 External Validity

This threat refers to how generalizable the results of the study are. In our case study, we applied data triangulation and interviewed seven employees of three different roles to get data from different sources. However, since we conducted the study in only one company, we cannot generalize the obtained results without further replication of the study which is discussed as future work in Section 7.8.

With regards to the tertiary review, the most recent publication was published 2014, which reviewed papers up to 2013. There is a chance that papers that propose newer solutions to our identified challenges have been published since then. However, since the multi-vocal literature review covers sources published up to August 2017, we could confirm the data extracted from the secondary sources.

2.9.2 Construct Validity

To minimize this threat we had to make sure that what we wanted to study (Challenges of establishing traceability) was understood by the participants of the study. To achieve this we first had a meeting with the two experts from the two departments where we explained the intentions of the study. In return, they also explained what their departments do. We also sent the interview guide and scope to the participants one week before the study. As mentioned in Section **??**, the interviews we conducted were not recorded due to legal matters but the interviewer took notes. To make sure that we did not misinterpret our findings, we showed our initial analysis to one of the senior experts for confirmation. This is known as member checking [180]. The multi-vocal literature review relies on publicly available sources, it is possible that it does not fully cover the state of the art in the automotive industry. In particular OEMs and Tier-1 suppliers do not make all information regarding their processes publicly available. We have tried to mitigate this thread by being as broad as possible in our search terms and include as many sources from different provenance as possible to construct a picture that is as complete as possible.

2.9.3 Reliability

To ensure that the results of a study are reliable it is important to make sure that the study can be repeated by other researchers and get the same results. While the settings of the interview cannot be replicated, the artifacts used such as the definition of the scope of the study and the interview guide were well documented and can be used for replication of the study. For the literature review, especially the MLR, even though we have documented our process and have traceability of which source produced which challenges and solutions, repeating the study to obtain 100% similar results is a challenge. This is because for the very short sources (e.g., blogs, presentations, forums), the information given is brief and therefore leaves room for interpretation. To reduce the chances of misinterpretation, two researchers went over the ambiguous challenges and solutions together to code them.

2.10 Related Work

Regan and colleagues [8], conducted a literature review to identify the barriers of traceability and their solutions from literature. In their work, they propose a framework which consists of the categories of the challenges and their solutions. Their framework is quite similar to the categories of challenges that we have proposed. However, their work does not investigate if these proposed solutions work in practice, which is something that our research does by complementing the literature reviews with an industrial case study.

Further related studies are those by Torkar et al. [9] and Cleland et al [34]. In [9], the authors performed a systematic literature review, with the aim of identifying requirements traceability definitions, tools, practices and challenges. They also complement their work with a case study in two companies. In their results, they give a list of challenges and how they are relevant for the two

companies. That study is similar to ours but their literature review only includes papers up to 2007 while ours includes studies of up to 2014. Also in their research the studied companies are not in the automotive domain but in the telecommunication domain and mobile applications domain. In [34], the authors reviewed four recent industrial studies and interviewed eight practitioners on traceability practices. The authors propose several research questions that need to be investigated in order to achieve the seven desired qualities of traceability proposed in [85]. These qualities are that traceability needs to be purposed, cost-effective, configurable, trusted, scalable, portable and valued. These quality attributes correspond to the findings in our study, for instance for traceability to be trusted, there needs to be methods for assessing the quality of links. Also in the study, one of the conclusions is that more collaboration with industrial practitioners and researchers is needed in order to ensure that the solutions from research are actually applicable in practice. Our study is an example of the research proposed here.

A study by Kannenberg & Saiedian [38] reviews the existing literature to investigate why software requirements traceability still remains a challenge. They conclude that manual traceability methods and existing tools are inadequate for the needs of the software development companies, a finding support by our investigation.

2.11 Conclusion

This paper provides an exhaustive overview of traceability challenges and solutions in the automotive domain and contrasts them with those found in general literature. Our study shows that there is a significant overlap between general challenges and solutions and those found in the automotive domain. It provides evidence that many solutions proposed in the literature are not applicable in the automotive domain due to its specific set of characteristics, such as system complexity, the safety-criticality of the developed systems, and the distributed development split between the OEMs and suppliers.

We used a tertiary literature review to explore general traceability challenges and solutions reported in literature, a multi-vocal literature review to elicit challenges reported in the automotive domain by different provenances such as tool vendors, consultants, academia and users, and a case study to explore how the challenges are experienced in practice.

While the tertiary review revealed challenges and solutions mostly from academia, the MLR was a richer data source (due to the diversity in the provenances). The MLR also gave an indication of which challenges are particularly prominent in the automotive domain. Challenges such as *Diverse artifacts and tools* and *Manual work*, e.g., were reported by all provenances. The same is true for solutions where, e.g., *Integrated tool platform*, *Tool integration* and *Automation* where reported by all provenances. The MLR also showed the difference in challenges that are mainly discussed in academia and those discussed with practitioners such as tool vendors, consultants and users.

The case study validated our findings as most of the challenges were found there as well. In addition, it revealed six unsolved challenges at the company: 1) Manual work of creating and maintaining traceability links, 2) Traceability

activities perceived as an overhead, 3) Lack of visualization tools, 4) Manual assessment of links, 5) Hard to measure the return on investment of traceability and 6) Lack of universal standards for exchange of traceability links.

There are proposals for solutions for most of the unsolved challenges. However, for the case we investigated, these solutions were either tried and did not fully solve the problem (e.g., an integrated tool platform to solve the diversity of tools problem) or the solutions could not be applied due to constraints that are specific to the automotive domain such as the requirement to follow safety standards like ISO 26262. This limits, for instance, the applicability of machine learning to generate links for safety-critical applications. Given that our static validation was conducted in one company, this is no indication that these challenges are also unsolved in other automotive companies. Nevertheless, we identify solutions that can be applicable to solve these challenges given the constraints found in the automotive domain. It is therefore still important to investigate how the proposed solutions in literature can be tailored and made applicable to this domain. In cases where tailoring of the solutions will not be enough, new approaches to solve these challenges can be investigated.

For future work, we plan to investigate how solutions proposed in Section 7.6 will be able to work in practice, by implementing and trying them with practitioners. As part of our research we have developed an open source traceability tool[8] that allows manual creation of links to arbitrary artifacts. In terms of the solutions found in our review, it addresses *Tool integration* and *Report generation*. Our concrete plans are to investigate how to combine automatically created links (for instance from model transformations) with manually created links. We will also investigate how to support users with semi-automatic maintenance of traceability links through notifications and collaborative features such as commenting on links. Furthermore, we will investigate how such a dedicated traceability tool can be integrated into the development process of a company. To contribute to the best practices of traceability, we also plan to work together with our industrial partners, mainly from the automotive domain, to provide different traceability information models for the different systems found in this domains. For instance we will provide information models for traceability when developing product lines and when developing multi-core systems.

Acknowledgements

Funding: This work was supported by Vinnova (grant number 2014-01271) as part of the ITEA^2 project AMALTHEA4Public.

Appendix A: Additional data about MLR

This appendix contains tables with additional information about the multi-vocal literature review.

[8] https://eclipse.org/capra

Table 2.4: Frequency of different source types in the MLR by provenance

Source Type	Total	Academia	Tool Vendor	Consultant	Mixed	User
Conference Paper	65	23	5	2	23	12
Workshop Paper	11	7	1	0	3	1
Journal Paper	12	3	1	1	6	1
Book Chapter	3	2	0	0	0	1
Peer reviewed	91	35	7	3	32	15
Book	3	2	0	0	0	1
Whitepaper	19	1	10	6	1	1
Presentation	35	6	11	11	3	5
Press release	3	0	2	0	0	1
Blog Entry	7	0	5	2	0	0
Thesis paper	3	2	0	0	0	0
Job posting	3	0	0	2	0	1
News article	6	0	4	2	0	0
Project Deliverable	4	1	0	0	1	2
Forum post	1	0	0	0	0	1
Course announcement	1	0	0	1	0	0
Tool description	43	0	41	2	0	0
Case description	8	0	7	1	0	0
Technical Report	2	1	0	0	1	0
Website	1	0	0	0	0	0
Service description	3	0	1	2	0	0
Magazine Article	8	0	4	3	0	0
Manual	2	0	0	0	0	0
Thesis description	2	1	0	1	0	0
Training Material	1	0	0	0	0	0
Talk abstract	1	0	0	0	0	1
Workshop Proceedings	1	0	0	0	1	0

Chapter 3

Paper B

Traceability Maintenance: Factors and Guidelines

S. Maro, A. Anjorin, R. Wohlrab, J.-P. Steghöfer

31st International Conference on Automated Software Engineering (ASE 2016), Singapore, Singapore, September 3-7, 2016.

Abstract

Traceability is an important concern for numerous software engineering activities. Establishing traceability links is a challenging and cost-intensive task, which is uneconomical without suitable strategies for maintaining high link quality. Current approaches to Traceability Management (TM), however, often make important assumptions and choices without ensuring that the consequences and implications for traceability maintenance are feasible and desirable in practice. In this paper, therefore, we identify a set of core factors that influence how the quality of traceability links can be maintained. For each factor, we discuss relevant challenges and provide guidelines on how best to ensure viable traceability maintenance in a practical TM approach. Our results are based on and supported by data collected from interviews conducted with: (i) 9 of our industrial and academic project partners to elicit requirements for a traceability tool, and (ii) 24 software development stakeholders from 15 industrial cases to provide a broader overview of the current state of the practice on traceability maintenance. To evaluate the feasibility of our guidelines, we investigate a set of existing TM solutions used in industry with respect to our guidelines.

3.1 Introduction and Motivation

Traceability can be defined as the ability to relate different artefacts created during the development of a software system. This also includes the ability to identify stakeholders that have contributed to the creation of artefacts, and the rationale that explains the need of these artefacts [5]. Traceability Management (TM) incorporates the creation, maintenance, and use of traceability links. It is an important concern that cuts across numerous domains and application scenarios including tool integration [181], requirements management (RM) [14], software product line management [182,183], model driven engineering [108,184, 185], and compliance with standards such as CMMI [186] and ISO 26262 [75].

All activities associated with keeping traceability links up to date and consistent are referred to as traceability maintenance. Traceability links rapidly become obsolete and effectively useless if they are not maintained as other artefacts evolve [29]. As the manual maintenance of links is error prone and expensive, a tool-supported approach to traceability maintenance is required if the benefits of traceability are to be realised. The main contribution of this paper are factors that impact traceability maintenance and guidelines on how to address them when designing TM tools. This contribution is based on an analysis of the spectrum of possible solutions extracted from interviews with industry practitioners. A further contribution is an overview of how the guidelines are realised in existing TM tools.

The promised benefits of traceability include improving the quality of software systems by supporting tasks related to maintenance, evolution, documentation, testing, and reuse. Traceability makes these tasks less dependent on individual experts and improves system acceptance by increasing understandability [5, 10, 85] A current challenge is, however, to *cost-effectively* enable these promised benefits [29]. To ensure this, it is of the utmost importance to guarantee that a high traceability link quality can be maintained in the face of changes to connected artefacts.

Traceability link "quality" is typically quantified by a combination of measurable properties including completeness, correctness, accuracy, precision, confidence, etc. [10]. These properties can only be defined precisely in a specific context and will thus be referred to collectively in the following as the general level of *consistency* of all traceability links. There are some automated approaches to maintain consistency, e.g., constraint-based [185], grammar-based [187], or based on machine learning techniques [188]. In practice, however, it often remains unclear why the particular chosen approach is feasible or desirable. The data from our interviews reveals two main challenges: (i) traceability links are still mostly created manually in practice as there is not yet sufficient trust in the quality of automation techniques, and (ii) one must cope with connected but highly *heterogeneous* artefacts across tool boundaries.

When designing and developing a TM solution that combines manual and automated traceability maintenance techniques, multiple factors must be considered. It is crucial to understand their consequences on traceability maintenance. Our aim is to provide a systematic set of guidelines that can be applied when establishing traceability maintenance as a crucial part of a practical TM approach. These guidelines are based on a set of primary factors that influence how the consistency of traceability links can be maintained.

For each factor, we discuss relevant challenges and suggest solution strategies together with their respective consequences and implications. These factors and guidelines constitute a novel contribution based on empirical evidence. Thus, the main research question for this paper is as follows:

What are the primary factors that affect how and to what extent a TM solution can provide traceability maintenance?

We conducted semi-structured interviews with 9 industrial practitioners to elicit requirements for a traceability tool, and with 24 additional software development stakeholders from 15 companies to provide a broader understanding of the practical challenges involved in establishing a viable and flexible TM solution.

The rest of the paper is structured as follows: In Section 3.2 we introduce basic terminology. Our main contribution, important factors and guidelines to consider when addressing traceability maintenance, is presented in Section 3.3. In Section 3.4 we evaluate the feasibility of our guidelines in practice by investigating a set of existing TM solutions with respect to our guidelines. Our paper concludes with an overview of related work in Section 3.5, threats to validity in Section 3.6, and future areas of research in Section 3.7.

3.2 Foundations

As TM is a task that cuts across multiple application domains and technological spaces, our terminology is chosen to be generic enough to incorporate both manual and informal TM strategies. Research on *bidirectional transformations* (bx) has many parallels to TM, including the central concept of consistency for a given set of artefacts, as well as the requirement to be as technology agnostic as possible. Our definitions are thus inspired by work on bx such as [189].

Let us refer to the "things" that we want to work with (modify, trace to and from) as *models*. We do not care what exactly models are (this can be very different from one domain to another), only that they can be modified to result in other models. Let us refer to such a modification as a *delta*. We are also interested in different "kinds" of models, which we shall refer to as *model spaces*. A model space basically groups together all possible states of a kind of model, connected by deltas. Again we do not care how such a model space is exactly induced as there are many ways to do this (using metamodels, constraints, grammars, etc.). Finally, we expect deltas to be composable and that it be possible to get from any model to any other model in the same model space. This is summarised succinctly in the following definition.

Definition 1. *(Model Space) A* model space $\mathcal{M} = (M, \Delta)$ *consists of a set* M *of models, a set* Δ *of deltas, and functions* $src : \Delta \to M$, $trg : \Delta \to M$ *that map a delta to its source and target model, respectively. For* $A, A' \in M$, *we denote* $a \in \Delta$ *as* $a : A \to A'$, *if* $src(a) = A$ *and* $trg(a) = A'$.

Every model space $\mathcal{M} = (M, \Delta)$ *is* connected*:* $\forall A, A' \in M \; \exists\, a : A \to A'$, *and* reflexive*:* $\exists\, id : M \to \Delta$, *a function mapping every model* A *to* $id_A : A \to A$, *a special* identity *delta. Finally, deltas can be composed via a binary operator* $; : \Delta \times \Delta \to \Delta$, *which is* associative: $(a; a'); a'' = a; (a'; a'')$, $\forall\, a : A \to A', a' : A' \to A'', a'' : A'' \to A'''$, *and for which identity deltas are* neutral: $id_A; a = a = a; id_{A'}$.

Example. As our running example we consider a software development project with (i) requirements in the ReqIF[1] format, (ii) implementation models as UML statecharts, and (iii) tests in form of C code. In terms of model spaces we thus have three model spaces: (i) $\mathcal{M}_{req}$ of all possible requirement models connected by deltas representing all possible changes (e.g., addition, deletion, and all means of editing a requirement), (ii) $\mathcal{M}_{uml}$ of all possible statechart models and deltas on statecharts, and (iii) $\mathcal{M}_c$ of all C programs and deltas on C programs. One could additionally restrict the model spaces to "well-formed" models, e.g., only considering C programs that compile, and requirements/statecharts that comply to the ReqIF/UML metamodel (and all constraints).

The concept of a "traceability link" is relatively difficult to fix, ranging in the literature from typed to untyped, binary to n-ary, and interconnected to isolated. In this paper, we choose not to define what a traceability *link* is but rather to view a distinguished model space as a special kind of *traceability model space* for connecting models from other model spaces. This means that traceability models and deltas are just as simple or as rich as any other models and deltas:

Definition 2. *(Traceability Model) A* traceability *model space is a distinguished model space* $\mathcal{M}_\tau = (M_\tau, \Delta_\tau)$*. Models* $T \in M_\tau$ *are referred to as* traceability models.

Example. For our running example, we take a model-based approach, defining a traceability model space $\mathcal{M}_\tau$ via a meta-model with an n-ary traceability "link type" connecting a requirement, multiple states and transitions in a statechart, and multiple tests (files with C code) together. We also decide to connect such traceability links with an association "isRelatedTo", effectively grouping related traceability links. The point here is that traceability models can be chosen to be just as rich as any other model.

The exact manner in which a traceability model "connects" a set of other models is also left open and can range from explicit edges (assuming a graph-based representation of models), implicit connections based on attribute values (IDs), and connections based on auxiliary structures such as tables, etc. For this paper, it suffices to introduce a *consistency function* that hides all such details, and decides how consistent a given set of models together with a connecting traceability model is. We choose to allow *levels* of consistency as opposed to "fully consistent" or "completely inconsistent", as a viable traceability maintenance solution should be able to cope with *partially* consistent models [190]:

Definition 3. *(Consistency Function) Given model spaces* $\mathcal{M}_1 \ldots, \mathcal{M}_n$*, and a traceability model space* $\mathcal{M}_\tau$*, a* consistency function *is a function* $R : M_1 \times \ldots \times M_n \times M_\tau \to [0, 1]$.

Example. Given model spaces $\mathcal{M}_{req}, \mathcal{M}_{uml}, \mathcal{M}_c$ and traceability model space $\mathcal{M}_\tau$ from our running example, a consistency function $R : M_{req} \times M_{uml} \times M_c \times M_\tau \to [0, 1]$ can be specified as a pragmatic combination of automated sanity checks and decisions to be made by a domain expert:

[1] Requirements Interchange Format (`omg.org/spec/ReqIF/`)

- *Validity:* $R(m_{req}, m_{uml}, m_c, m_\tau) := 0$ if m_τ is invalid, i.e., does not conform to its metamodel. This means that "broken" traceability links (wrong types, violated multiplicities, etc.) are not to be tolerated.
- *Completeness and Correctness:* if m_τ is valid, then $R(m_{req}, m_{uml}, m_c, m_\tau) := 0.2 \cdot comp + 0.8 \cdot corr$, where *comp* is the number of "covered" requirements, i.e., requirements connected to a traceability link, divided by the total number of requirements, and *corr* is the number of correct traceability links divided by the total number of traceability links. Correctness of a traceability link is manually determined by consulting a domain expert.

Note that this consistency function penalises incorrect links more than missing links, and can be extended analogously to handle uncovered elements also in the statecharts and tests.

Given that it is possible to gauge the consistency of a set of models connected by a traceability model, we can now define the task of *traceability maintenance*. The central idea is to define a traceability maintainer on *deltas* instead of just models, i.e., to supply information on how a set of models has evolved, together with the old traceability model. The task of traceability maintenance is then to compute a suitable delta on the traceability model. This is depicted schematically in Fig. 3.1 as "completing the square" and is subsequently formalised in Def. 4. This differs from general consistency restoration, where the input deltas can also be manipulated [189]. In the case of traceability maintenance, the expectation is that *only* the traceability model is changed.

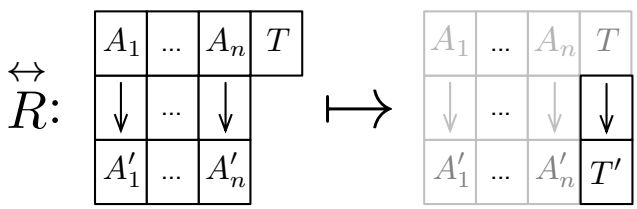


Figure 3.1: Completing the square

Definition 4. *(Traceability Maintainer) Given model spaces $\mathcal{M}_1 \ldots, \mathcal{M}_n$, and a traceability model space $\mathcal{M}_\tau$, a* traceability maintainer *is a function* $\overleftrightarrow{R}\colon \Delta_1 \times \ldots \times \Delta_n \times M_\tau \to \Delta_\tau$.

Example. For the consistency function R defined previously, the following represents a traceability maintainer $\overleftrightarrow{R}$:

[a] Delete all broken traceability links (fully automatic).
[b] Request a review of all traceability links by a domain expert to evaluate correctness, supplying exactly what was changed as input. Incorrect traceability links that cannot be fixed should be deleted (manual).
[c] Determine uncovered requirements and ask a domain expert to add missing links (semi-automatic).

The assumption in Step (2), which is reasonable but does not hold in general, is that a domain expert (or some software component if this step is automated) does not need to review *all* links if provided with detailed change information.

A basic property of a useful traceability maintainer is that the maintainer either improves (or retains) the current situation, or does nothing at all. This

expectation holds for manual and fully/semi-automated traceability maintenance alike; if manipulating the traceability model worsens the current situation then the changes are not worth applying.

Definition 5. *(Consistency Improving) For model spaces* $\mathcal{M}_1 \ldots, \mathcal{M}_n$*, a traceability model space* $\mathcal{M}_\tau$*, and a consistency function* $R : M_1 \times \ldots \times M_n \times M_\tau \rightarrow [0, 1]$*, a traceability maintainer* $\overset{\leftrightarrow}{R}$*:* $\Delta_1 \times \ldots \times \Delta_n \times M_\tau \rightarrow \Delta_\tau$ *is* consistency improving *if* $R(A_1, \ldots, A_n, T) \leq R(A'_1, \ldots, A'_n, T')$*, where* $\delta_1 : A_1 \rightarrow A'_1 \in \Delta_1, \ldots, \delta_n : A_n \rightarrow A'_n \in \Delta_n$*, and* $\delta_T = \overset{\leftrightarrow}{R}(\delta_1, \ldots, \delta_n, T) : T \rightarrow T' \in \Delta_\tau$*.*

Further properties concerning, e.g., how "much" of the traceability model is changed (the assumption being that "smaller" changes are preferred), can be specified. The interested reader is referred to, e.g., [191] for a related discussion.

3.3 Influential Factors and corresponding Guidelines

This section presents our findings and discussion based on data collected from the following sources:

(S1) Two focus groups[2] aimed at identifying traceability problems and collecting requirements for a traceability tool from both industrial and academic partners in the Amalthea4public project. The first session was with 5 partners from 2 companies developing embedded systems for forest automotives in Sweden and the second session was with 3 academic partners from 2 universities and one industrial partner (automotive supplier) from Germany. The collected traceability requirements were later refined through phone calls with project partners outside Sweden, and face-to-face meetings with one industrial partner in Sweden.

(S2) Semi-structured interviews[3] with 24 software development stakeholders from 15 industrial cases in Germany and Sweden. This was part of a larger case study aimed at investigating general traceability management practices in industry [192]. For this paper, we only use data related to traceability maintenance collected from the study.

The majority of the cases (cases 1-6) are from the automotive domain, followed by the domains of software development (cases 7-8) and telecommunications (cases 9-10). Other domains are IT services (Case 11), banking self-service automation (Case 12), electrical equipment (Case 13), embedded systems (Case 14), and industrial automation (Case 15). All interviewees had working experience of at least one year in their current roles, including development managers, quality managers, system software architects, and product managers. The interviewees worked in projects varying in size, from four to more than one hundred employees.

The interviews from S1 and S2 were recorded and transcribed. The data was used in several analysis sessions with four researchers to identify key factors and guidelines. We conducted cross case analysis to examine differences between

[2] http://tinyurl.com/jotqagy
[3] http://tinyurl.com/ht2hmzk

cases and identify practical needs. In the following, we present our findings, referring to respective sources to support our arguments.

3.3.1 Factor 1: Versioning

A realistic application scenario will involve *multiple users* working together more or less concurrently on a common set of models. This implies that a *Version Control System* (VCS) of some kind is most probably already present and in use. The conclusion that versioning is a primary factor to consider is supported by (S1), as our project partners require explicitly that versioning be addressed appropriately in a proposed TM solution, and by (S2) as change propagation and VCS solutions play an important role in *all* 15 cases.

The effect of versioning on traceability maintenance can be explained by considering the possible input to a traceability maintainer, depicted in Figure 3.2 with labels ❶, ❷, ❸. If versioning information is completely ignored ❶, traceability maintenance becomes relatively challenging as the maintainer is in effect presented with some versions (perhaps the latest versions) of all models and is expected to update the traceability model. Without any provided deltas, however, a maintainer can only assume that everything was created from scratch meaning that all models, including the traceability model, must be fully inspected. This corresponds to a so called *batch* or non-incremental scenario, well-known from research on model synchronisation (cf. [193] for a classification of application scenarios for model synchronisation). This situation is problematic as it is difficult, if not impossible, to guarantee consistency improvement in any way (the consistency of the previous "state" cannot be determined as previous versions are unavailable). We are ready to formulate our first guideline concerning versioning:

(G1) *Version your traceability model just like all other models, especially ensuring that it is included in any consistent tags (beta, release, etc.). Strive to provide the same level of support and integration with your VCS for your traceability models as for any other models.*

Ample support for (G1) is provided by (S2) as the majority of our interviewees explicitly stated that it would be beneficial to have a versioning solution for traceability models. In some cases (3 of 15), a traceability model is indeed versioned, connecting models in specific versions. This is stated by a software architect from Case 2 to be a major advantage as "correct" traceability links do not get "automatically incorrect", but rather "outdated".

We interpret this as meaning that the task of traceability maintenance is

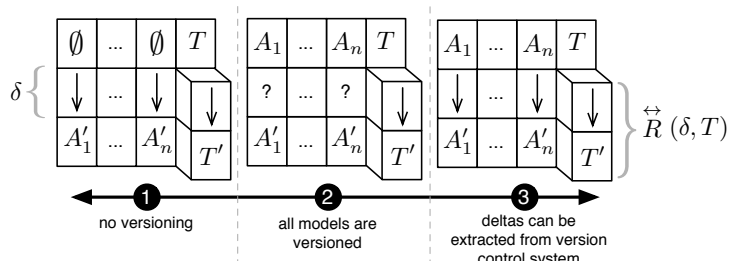


Figure 3.2: How versioning affects consistency

well-defined in the sense that a traceability model T can be evaluated with respect to the correct set of connected models without imposing automatic updates, i.e., forcing a potentially problematic evaluation of T in the context of $A'_1, \cdots, A'_n$ (cf. ❶ in Figure 3.2).

Even if all models are versioned and tagged together with the traceability model, the task of traceability maintenance remains challenging if explicit deltas are not provided. This situation is depicted as ❷ in Figure 3.2 and corresponds to a so-called *state-based* scenario, where the traceability maintainer is only provided with the previous and current versions of all models, and must somehow determine the missing deltas (depicted as question marks in the diagram). This is better than the batch case, but is still suboptimal as the deltas have to be determined, e.g., by comparing the two versions. This is problematic as it entangles consistency improvement with delta recognition, two difficult tasks that should best be handled and controlled separately [189]. This brings us to our second guideline on versioning:

(G2) *Ensure that you are able to extract explicit deltas for all models from your chosen VCS.*

Following this guideline means that you are able to provide all deltas required for traceability maintenance. This is depicted as ❸ in Figure 3.2, representing the ideal situation required for traceability maintenance as all deltas are present.

Guideline (G2) is supported by (S2) as multiple interviewees describe their expectations of how a traceability maintainer should work as follows: a maintainer must check if there are implications caused by *evolving connected models*. If a versioning solution exists, the traceability model must be appropriately *updated with respect to the new versions* of the models, i.e., one must decide if there should (still) be a link or not. Furthermore, the vast majority of interviewees attributed the most common source of inconsistencies to missing "deltas" and corresponding change propagation. For example, a software architect from Case 2 stated that most inconsistencies are probably introduced when performing changes (changing a signal) for which no information is provided about what is connected and potentially affected (e.g., connected requirements). The point here is that without explicit deltas, the entire traceability model must be inspected, regardless of if the consistency maintainer is fully automated, semi-automated, or manual. Even in an optimal situation with all deltas, there is still no general guarantee that necessary updates to the traceability model are "local" in any sense, but the chances of providing an "incremental" and more efficient traceability maintainer are increased.

3.3.2 Factor 2: Tool Boundaries

A typical application scenario for traceability will involve multiple model spaces and often many tools, with which the different models are managed. Planning the scope and boundaries of a TM tool is, therefore, a crucial factor that has a substantial impact on traceability maintenance.

Our requirements (S1) show that project partners require integration of different VCS approaches, RM tools, development environments, and modelling standards, to name just a few of the models spaces and tools involved.

Our interviews (S2) show an equally wide variety of stakeholders from several disciplines, often organised in separate departments with several tools. 14 of our 24 interviewees stated that the heterogeneous nature of the used tools makes traceability management in general, and traceability maintenance in particular, quite difficult in practice. Due to inadequate tool integration, it is often difficult to establish connections between models stored in different tools.

In some cases, workarounds are provided using the manual mapping of IDs (such as in Case 12 or Case 10). In Case 11, the developers reference the requirements specification with the current version number as a source code comment. Such ad-hoc solutions negatively impact traceability maintenance: the relevant interviewees confirmed that the traceability links established in this manner can only be managed manually and can get easily outdated. It is thus important to plan for a heterogeneous tool landscape, with an understanding of how this impacts traceability maintenance.

The range of choices is depicted schematically in Figure 3.3. On one end of the spectrum ❶ is a *holistic* tool environment that directly supports all relevant tasks, including traceability management. Everything is fully integrated in essentially the same tool. On the other end of the spectrum ❸ is a separate TM tool that only manages traceability models and must establish links to models managed by other tools.

In between these extremes are hybrid solutions ❷ where a mix is chosen. While eliciting requirements from our project partners (S1), we were confronted with contradicting requirements: some partners were interested in traceability to and from primarily requirement specifications and thus suggested that the envisioned TM solution incorporate direct support for RM. This was essentially demanding a hybrid solution combining TM and RM in the same tool. Other partners, however, were already using established RM tools and ruled out changing or switching to a new RM tool.

Interviewees from 6 of our 15 cases from (S2) state that interorganisational collaboration would benefit from using a common tool. For example, a developer from Case 1 stated that it would be helpful to have a common platform to communicate with suppliers. The problem is that there is no standard way to communicate. Different suppliers work with different tools and approaches (e.g., document-based, model-based). With a common tool or platform, one would not have to worry about different standards.

Although a holistic environment might work in some cases — e.g., for smaller companies such as in Case 14, where the collaboration with customers

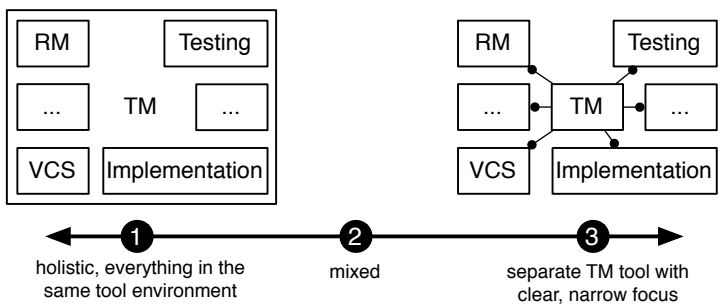


Figure 3.3: Setting tool boundaries

is very close and they can get direct access to development artefacts — in many scenarios this will not be feasible. In Case 6, for example, external collaboration is accomplished using the export of specification documents and e-mail exchange. It was stated that due to legal and intellectual property issues, it is infeasible to use one central platform with other organizations. Even in supposedly holistic solutions, such as Case 13 and Case 9, some traceability links still exist that point to bugs or issues reported by the customer and are thus "external" to the tool.

The Head of Software Quality Assurance from Case 12 points out that having an "orthogonal tool that only takes care of traceability" ❸ might be the solution. Such a tool would need to have good interfaces to existing tools (for RM, test management, etc.) and allow the creation of links between models managed by different tools. Our next guideline, therefore, encourages TM solutions to provide direct access to their managed traceability models, as this opens up the TM tool, simplifying integration with external tools:

(G3) *Provide well-defined interfaces and easy, direct access to managed traceability models.*

Holistic and separate TM solutions both have advantages and disadvantages, many of which were stated by our interviewees. With respect to traceability maintenance, a holistic tool environment is able to guarantee a certain minimal consistency, e.g., by forbidding changes that break traceability links, and by forcing the user to first of all delete or adjust affected traceability links before making changes that might cause inconsistencies. One could also weave traceability link creation into a given process supported by the TM tool, ensuring that certain traceability links are created *eagerly*, i.e., "captured" immediately at certain steps in the process. This is advantageous and significantly simplifies the task of traceability maintenance. Establishing such a holistic tool and acquiring adequate acceptance for it is, however, challenging, especially in the context of a multi-partner, open-source project such as Amalthea4public.

For a separate TM tool, every model apart from the traceability model is external in the sense that the TM tool does not control where these models are persisted and how/when they are changed. To enable this, a strategy is required to connect elements in such external models to elements in the traceability model. Possible solutions include establishing *proxies* for these elements, whose creation and maintenance are handled by corresponding (tool) adapters. With respect to consistency, this means that the maintainer has to be able to deal with traceability links that can become broken due to changes (no minimal consistency can be guaranteed), as well as support the *delayed* creation ("recovery") of new traceability links due to changes. This can be substantially more difficult and less scalable than in a holistic situation.

Mixing these strategies, however, tends to amplify the disadvantages, leading to a situation where there is an internal concept not only of traceability models, but also of, for example, requirement models. This means that some import/export mechanism must be provided to get existing requirements specification into the TM tool, paired with the possibility of linking to external models in other tools (e.g., implementation and test models). This results in a complex and potentially confusing workflow, where some models are treated differently than others. In the worst case, imported models might still be changed externally, demanding some additional form of update or

synchronisation mechanism. Our guideline in this respect is thus as follows:

(G4) *Aim for either a holistic solution, or a completely separate TM tool with a carefully designed tool adapter concept. Avoid combining both strategies.*

Support for (G4) is provided by our interviews (S2): interviewees from 9 of 15 cases stated that having one common platform across disciplinary borders would be beneficial.

To address the challenges of establishing a separate TM solution and numerous tool adapters, we suggest:

(G5) *Use a common standard and/or technological space as "glue" to simplify the development of tool adapters.*

This is supported by (S2) as, for example, a product manager from Case 5 and a system software architect from Case 6 state that having one tool for all tasks is not feasible but that one should rather try to achieve traceability using better interfaces across tool boundaries (such as OSLC[4] or EMF[5]).

3.3.3 Factor 3: Configurable Semantics

The types and semantics of traceability links vary depending on the domain. It is therefore impossible to implement *generic* but still adequately useful TM consistency functions as consistency is defined based on the type and semantics of the links. This means that "consistency" must be defined and tailored to each domain, possibly even to specific processes, companies, and projects. Defining consistency functions and corresponding traceability maintainers is thus a central and recurring task for TM and should be supported as much as possible by any TM approach.

Support for regarding such diversity as an important factor is provided by (S1): Different project partners require different traceability link types. For instance, partners having a product line approach express the need for links related to variability management while those developing multi-core systems require traceability links for task mapping purposes. Such diverse consistency-related requirements cannot be addressed with a single, fixed definition of a traceability model space, indicating that a suitably flexible configuration process is crucial.

The data collected from our interviews (S2) also reveals a desire for diverse semantics of links e.g., "satisfies", "transferred from", and "refers to" mentioned by a quality manager from Case 7 or "verifies" and "fulfils" mentioned by a system software architect from Case 6, to mention a few.

As motivation for configurable semantics, our interviewees stated: (i) improvement of traceability maintenance (the Head of Software Quality Assurance from Case 12), (ii) to support understandability, especially for new developers (a developer from Case 1), (iii) to simplify reviews and creation of status reports, especially for large models (a project manager from Case 14), and (iv) to enable better search and filter functions (a system software architect from Case 6).

A factor that greatly influences the complexity of consistency functions and corresponding maintainers is the degree to which the traceability model space captures domain-specific semantics and whether the links are explicitly

[4] `http://open-services.net/`
[5] `https://eclipse.org/modeling/emf/`

stored or exist implicitly based on, e.g., naming conventions. The spectrum of choices ranges from traceability model spaces without any domain-specific semantics at all (any connection is possible), to traceability model spaces with a rich domain-specific semantics (connections are restricted to only what makes sense). The former simplifies TM tool support but shifts all complexity to the consistency function and maintainer, while the latter captures some level of consistency already in the traceability models, thereby simplifying corresponding consistency functions and traceability maintainers.

To discuss the effect of implicit vs. explicit semantics on consistency maintenance in more detail, Figure 3.4 depicts the range of choice divided into implicit links ❶, e.g., based on conventions, and explicit links, which are further divided into generic ❷, fixed ❸, and domain-specific ❹.

Implicit links ❶ are connections between traceability models and other models based on conventions such as naming schemes, identifiers, etc. For example, when committing code for a bug fix into a VCS, the ticket number of the bug report should be written in the commit message. Such conventions are problematic as they can be hard to enforce and are often regarded only as "best practice" leading to numerous violations or alternative and possibly conflicting conventions.

Implicit links can be very difficult to check for programmatically, e.g., if the referenced fixed bug is described textually (in a manner that is clear for a human reader) instead of entering its unique ticket number (similar examples could be observed in Case 10, Case 11, and Case 12). Explicit links, represented by elements in the traceability models, are easier to analyse and keep consistent. This brings us to our first guideline on configurable semantics:

(G6) *Avoid implicit, convention-based traceability links and strive instead for explicit links that can be checked with tool support.*

Explicit links can vary substantially regarding the degree to which domain-specific semantics can be captured. *Generic* links ❷, are all of the same basic "type" and can be used to establish connections between anything. This is advantageous for two reasons: (i) it is easy to provide generic tool support, and (ii) such links are flexible in the sense that connections can be established even in unforeseen situations.

From the point of view of consistency maintenance, however, almost all complexity is shifted to the consistency function and maintainer, which have to determine consistency based on the context of established connections. This is not only challenging but can even be impossible in some cases, if there is not enough context information present to retrospectively determine what such a generic link actually means. The disadvantages of generic links can be addressed by allowing additional meta-information to be embedded in links.

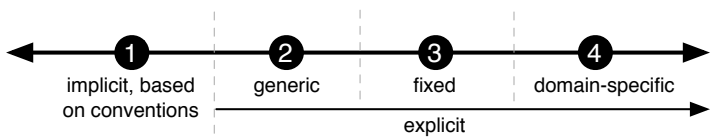


Figure 3.4: How rich are your traceability models?

In an attempt to retain the advantages of generic links with respect to generic tool support, a common strategy is to provide a rich, but *fixed* traceability model space ❸. This means that the TM solution provides, e.g., numerous attributed link types and relations that are, however, fixed with no or only very limited possibilities of extension. This is certainly an improvement over implicit or generic links, but the fixed model space might be either too complex or not rich enough for a certain domain, process, or company. In such a case, complexity is again shifted to the traceability maintainer as it is impossible to embed the required semantics into traceability models. The advantage of this approach is that the TM solution can provide a substantial amount of functionality and consistency checking out-of-the-box.

Finally, explicit links can be completely *domain-specific* ❹ if the TM solution allows the underlying traceability model space to be swapped. This enables domain-specific semantics to be represented, e.g., by appropriate attributes, types and relations. The advantage of this approach is that the TM solution can be adapted to a wide range of domains and applications without sacrificing semantics. Traceability model spaces can be chosen to be very rich, making it impossible, e.g., to create "wrong" links. A challenge with this approach is that generic functionality provided by a TM solution is limited. Substantial effort must be spent on re-implementing domain-specific parts, in particular in relation to traceability maintenance. In practice, therefore, some aspects are typically fixed such as the general "shape" of a traceability link. This discussion is summed up in the following guideline:

(G7) *Prefer domain-specific, semantically rich traceability model spaces as this simplifies traceability maintenance.*

Support from our interviews (S2) is provided by a quality analyst from Case 4, who describes the current usage of generic links as "immature" and "work in progress", and would prefer to be able to attach more semantics. An interesting observation is made by a product manager from Case 5, who mentions that it is virtually impossible to get the exact semantics perfectly right at the start of a project. The semantics must thus be adapted and updated continually during the lifetime of the project, not only by adding new "types" of links, but also by refining and even deactivating existing types. Concerning tool boundaries, the Head of Test Management from Case 7 states that especially links to external tools should be as "rich" as possible. A quality manager from Case 7 describes the current usage of a commercial TM tool with a fixed semantics as unsatisfactory; the TM tool "knows nothing" about extra semantics that users in the company have decided upon and that (hopefully) everyone in the company is aware of and adheres to. Such a fallback to relying on conventions has of course similar disadvantages as using implicit links.

3.3.4 Factor 4: Consistency Specification

Our final factor concerns *how* consistency is specified and consequently maintained. Support for considering this as a primary factor is provided by (S2), as numerous interviewees expressed the need to establish trust in the consistency of traceability links. For example, the Chief Technical Officer from Case 8 stated that the quality of traceability links must be so high that their benefit becomes

obvious to all stakeholders. If the stakeholders do not trust the traceability links, then they will not be used, and will not be improved.

The solution space for consistency specification as depicted in Figure 3.5 is spanned by two orthogonal dimensions: a dimension concerning the manner in which traceability maintainers are applied (the vertical axis), and a dimension characterising the possible classes of their underlying consistency functions (the horizontal axis).

From our interviews, we have identified two main strategies of maintaining consistency: a top-down, process-oriented, mostly eager (consistency violations are fixed immediately) strategy ❶, and a bottom-up, ad-hoc, mostly delayed (consistency violations are fixed only on-demand) strategy ❷. Both strategies appear to be equally successful in practice and are often mixed, with the choice mainly depending on the primary users of the TM solution. We thus suggest the following guideline for this dimension:

(G8) *Ensure that your TM solution supports a flexible combination of both top-down and bottom-up strategies.*

Support for this guideline is mainly provided by (S2): 7 of our 15 cases all have a strong focus on requirements management when it comes to applying traceability. All of these cases except Case 9 use a dedicated RM tool and already organise requirements and their breakdown with it. Integration and system test cases are usually also stored in the tool (or a connected test tool) and linked to the requirements. Many of these cases are part of bigger organizations, in which several companies work together on projects where safety and quality certification is often highly relevant. This demands organised, fixed processes and clear responsibilities. In Case 7, for example, formal reviews are conducted before a milestone is completed. In Case 9, connections between requirements and the design of a measurement system are recorded in spreadsheet files and documents, following a strict and well-defined process. In all these mainly requirements-centred cases, TM in general, and traceability maintenance in particular, is handled in a top-down fashion initiated by project management. Often, but not always, consistency is maintained as part of a defined process, e.g., creating or updating traceability links immediately after the connected artefacts are created or changed in a certain step.

Many other cases feature a more developer-driven, ad-hoc approach. In

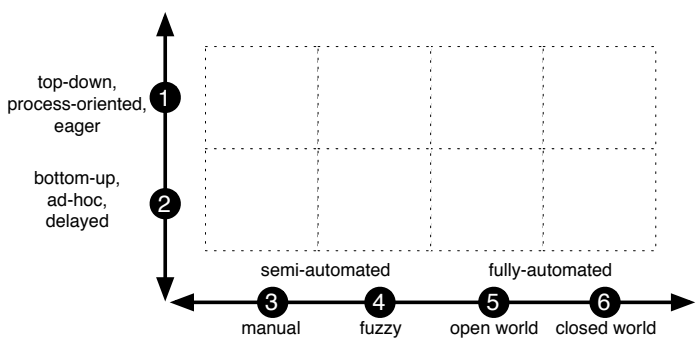


Figure 3.5: How is consistency specified?

these cases, traceability links are created and updated on-demand by developers, while focussing on software implementation. This typically involves a software configuration management system handling source code management, tracking of defects (issues, bug reports), and the management of implementation tasks (tickets) in a project. These cases typically drive their development based on implementation tasks, and require traceability, e.g., from source code commits to tickets.

There exist some cases that cannot be assigned to exactly one of the two approaches mentioned above. This involves attempts to handle TM in a structured, top-down way — and at the same time, bottom-up approaches invariably arise from the developers' side. In 3 cases, both approaches are clearly present and coexist harmoniously and complementarily. In Case 15, for example, there exists both a requirements-centred approach to connect features and test cases, while a more developer-driven approach is used to link bugs or defects to the respective code commits.

Coming back to the horizontal dimension of Figure 3.5, consistency is in practice often defined manually ❸, i.e., involving a domain expert who has to decide to what extent a given traceability model is consistent or not. As this is, however, quite expensive, considerable research has been conducted with the goal of (fully) automating this task, e.g., [187, 188].

Some of these approaches are fuzzy ❹, i.e., similarity metrics and machine learning techniques are applied to detect patterns and suggest probable connections [188]. Such techniques are often combined with a manual approach resulting in a semi-automatic approach, providing support for a domain expert to make the final decision.

Although semi-automatic approaches already improve the situation, a fully automated approach has the advantage that traceability links can be regarded as "derived", i.e., created on demand and never persisted. This avoids inconsistencies completely by simply re-creating all links as soon as any changes are made. To achieve scalability in cases where this is necessary, numerous incremental and caching strategies can be applied to determine the (potentially) affected set of traceability links and avoid updating or re-creating all links. An underlying assumption that might be necessary, however, is that all traceability links can be (re)created automatically given the current state of all models.

From this discussion, it would appear as though fully-automated strategies are to be clearly preferred; we suggest, nonetheless, taking a *hybrid* solution instead:

(G9) *Support an integrated mix of manual and complementary automated approaches to consistency specification.*

This pragmatic guideline is supported by (S2), indicating that there is simply not yet enough trust and acceptance for full automation. As a quality analyst from Case 4 puts it, automatically creating and updating traceability links is a difficult task; some form of validation or evidence is required to convince users that such traceability links are actually consistent. Especially interviewees from the automotive domain state that it would be very disturbing to have inconsistent traceability links in a system. A manual inspection of traceability links might be expensive, but at least avoids a false sense of high-quality traceability links.

Other interviewees are more open, stating that it would be interesting to improve traceability maintenance via automation strategies; they would, however, still use this more "as an input for a final manual step", as stated by the Chief Technical Officer from Case 8.

There is, at the same time, a clear wish for more automation expressed by numerous interviewees: based on clear naming and structural rules, the Head of Test Management from Case 7 would have no problem maintaining at least a subset of the traceability links automatically. In fact, a team leader from Case 3 mentions basic automation as an important point; simple rules based on steps in a well-defined process *should* ideally be automated to avoid tedious, repetitive, and manual TM-related tasks.

A well-accepted pragmatic strategy of how to combine manual and automated consistency maintenance appears to be the concept of *suspect links* [29, 194]. A product manager from Case 5 states that this technique of applying automatic consistency checks to identify "suspect links" is applied by numerous TM-related tools. Such suspect links are presented to the user for a final decision, possibly together with a set of standard "quick-fix" maintenance strategies that can be applied at the click of a button.

Fully-automated consistency specification approaches can be broadly classified into adhering to either the *Open World Assumption* (OWA) ❺, or the *Closed World Assumption* (CWA) ❻. In an OWA approach, traceability links that the traceability maintainer cannot classify as inconsistent are assumed to be consistent and retained. The maintainer is considered to be incomplete and the language of consistent traceability models is assumed to initially include all possible traceability models (everything would be accepted without a maintainer), and is progressively *restricted* as necessary. Suitable OWA strategies include constraint-based approaches, i.e., providing a set of constraints that must not be violated by any consistent link [185]. Links that are not referenced in any constraint are by default consistent.

In a CWA approach, links that cannot be classified as consistent are assumed to be inconsistent. The maintainer is considered to be complete and the language of consistent traceability models is assumed to be initially empty (everything would be rejected as inconsistent without a maintainer) and is progressively *extended* as necessary. Suitable strategies include generative, grammar-based approaches, i.e., a set of rules that generate all consistent traceability models [187]. Links that cannot be recognised by the maintainer as consistent are by default inconsistent.

Although there are numerous studies on successfully applying machine learning techniques to traceability management and maintenance [188], based on our requirements and interviews, we tend towards CWA approaches:

(G10) *For automatically generated links, prefer no links at all to (possibly) inconsistent links.*

This final guideline is certainly contentious and might not be valid in every application domain, but in the automotive domain and for the development of embedded and safety-critical systems, stakeholders appear to demand both a high confidence in traceability links and a zero tolerance for (possibly) inconsistent traceability links. As a team leader from Case 13 aptly states, users that encounter inconsistent traceability links tend to be utterly confused

by connections that do not make sense at all. Having no link at all is actually better than having an inconsistent link — in addition to eventually having to search in some other way for the desired connection, an inconsistent link forces you to first evaluate and conclude that it is indeed inconsistent and unhelpful.

3.4 Strategies in existing TM tools

We now discuss strategies for traceability maintenance employed by three TM tools currently used in industry. The discussion, structured with our proposed guidelines, is based on semi-structured interviews[6] with expert users (in one case) and the developers of the tool (other two cases).

3.4.1 Rational DOORS

IBM Rational DOORS,[7] also known as DOORS Classic, is a requirements management tool that is widely used in industry. It offers traceability features that allow tracing to different types of requirements, e.g., customer requirements, system requirements, software requirements, etc.

Versioning. Both requirements and traceability links are versioned (stored in a database) and can be included in tags. Deltas on requirements are recorded and are available to users. When an artefact connected by a traceability link changes, the user is informed and the delta is presented to the user, who should decide how to update the traceability model. The tool thus adheres to both (G1) and (G2).

Tool Boundaries. As the core functionality of DOORS is RM, traceability links from requirements to requirements are supported out-of-the-box. To address linking to model spaces other than requirements, DOORS provides an OSLC adapter for accessing the requirements. This is currently problematic, as changes to the models in external tools cannot be detected via such OSLC links. The clients of the OSLC adapter provided by DOORS also need to be implemented for each tool which takes substantial effort. Guideline (G3) is followed as links can be accessed via plugins/addons. DOORS, however, does not strictly follow guideline (G4) as it combines RM and TM. Although OSLC has its limitations, (G5) is followed by using OSLC as a common standard for linking to external tools.

Configurable Semantics. The traceability links created with the tool are explicit links, adhering to (G6). Semantics can be configured to a certain extent, as new link types with attributes and restricted source and target types can be created but, for example, n-ary links are impossible. Guideline (G7) is thus followed but with limitations.

Consistency Specification. DOORS allows both top-down and bottom-up consistency management strategies, adhering to (G8). Semi-automatic traceability maintenance is supported by the use of "suspect link detection", i.e., marking a link as "suspicious" as soon as one of the models it connects changes. This is well in accordance with (G9). In addition to manual links, DOORS supports the automatic creation of links for some generated models.

[6] `http://tinyurl.com/htvusac`
[7] `http://www-03.ibm.com/software/products/en/ratidoor`

For instance, test case skeletons (in form of Excel sheets) can be generated with a link back to the requirements the tests originated from. As the Excel sheets are, however, maintained manually after the generation step they can become inconsistent over time. This means that (G10) is adhered to, but without a viable means of maintaining consistency.

3.4.2 SystemWeaver

SystemWeaver[8] is a commercial holistic information management solution that aims to support the entire development life-cycle for software and systems engineering. SystemWeaver supports traceability by providing a means of connecting elements of models that reside within the tool.

Versioning. Every model in SystemWeaver is versioned and stored in a common database. The tool has its own VCS and is able to keep track of and provide all deltas. When a model is changed, SystemWeaver checks for any traceability links that are connected to modified elements and imply a potential inconsistency. If such links are found, the relevant deltas are presented to the user who decides if the link is to be updated to point to the newer versions of the models.

With respect to (G1), traceability information is not stored and versioned in an independent traceability model, but as part of the models residing in the tool. The versions of the models containing traceability links implicitly reflect the versions of the traceability links. As the tool implements its own VCS for all models residing in the tool and is able to provide explicit deltas, it adheres quite well to (G2).

Tool Boundaries. SystemWeaver is clearly a holistic tool, adhering to (G4). For models residing in the tool, it is relatively easy to keep track of what has been changed and propose corresponding changes to affected traceability links. It is also possible to configure workflows to prompt the user to create traceability links when certain model elements are created. Since traceability links are parts of the models in the tool, there is a need to traverse the models to get an overview of all traceability links. The tool provides visualization features out of the box and the user is able to get an overview of all the links. Guideline (G3) is thus partly followed. The tool is meant to be sufficient on its own, but some of its customers use it with other tools such as simulation tools. In such cases OSLC is used as glue technology for integration as suggested by (G5). According to an application engineer from the team developing SystemWeaver, ensuring and maintaining consistency to external tools requires a substantial amount of effort. This is done by creating tool-specific adapters that are typically not reusable.

Configurable Semantics. SystemWeaver provides considerable flexibility as metamodels can be used to configure the tool to a particular domain or project. By enabling explicit and domain-specific links, the tool is well in line with both (G6) and (G7). According to SystemWeaver's fixed meta-metamodel, however, the concept of a "connection" is defined as having a single source and a maximum of two targets. An application engineer of the tool stated, however, that this does not appear to be a major limitation for most use cases.

[8] `http://www.systemweaver.se`

Consistency Specification. SystemWeaver allows the definition of workflows enforcing when links should be updated, as well as ad-hoc link creation. This adheres to (G8) as both top-down and bottom-up approaches are possible.

The tool was originally designed for manual link creation. It is, however, also possible to define rules controlling when and how traceability links are created. Maintenance of links is semi-automatic; when a change is detected in a model element that has a link, the link is flagged as a "suspect link" and the user has to resolve this manually. This is in coherence with (G9). For (G10), as the tool is configurable, it is up to the final user to decide on suitable mechanisms for automatically generating links.

3.4.3 YAKINDU Traceability

YAKINDU Traceability (YT)[9], is a commercial, Eclipse- and EMF-based TM tool. It is a dedicated traceability tool.

Versioning. Traceability models in YT are EMF models, which can be persisted as XML files and thus versioned using any standard VCS. The tool strives to follow (G1) by providing extra diff and merge procedures implemented specially for traceability models.

Concerning (G2), however, version and delta information of the models connected by a traceability link must be obtained from the VCS that is used to store these models. The quality and availability of the deltas thus depend on the chosen VCS. For file types where version and fine-grained delta information cannot be accessed, YT computes a version based on the content of the model. If the model changes, YT analyses the current model and the information stored in the traceability link. If they no longer match, the user is prompted to update the traceability link.

Tool Boundaries. Traceability models are EMF models and can be accessed and used for activities such as impact analysis and change management. This correlates with (G3). YT is a dedicated TM tool, thus adhering to (G4).

All models apart from the traceability model are handled as external models by the tool. To create traceability links to these external models, an EMF representation is required. For non-EMF models, this is handled by tool adapters that create EMF representations of models from different tools. A tool adapter makes a specific type of file format available to YT, for instance, an Excel adapter makes Excel files (up to cell level) available to the tool, while a DOORS adapter makes DOORS requirements available to the tool. This follows (G5), as EMF acts as "glue" technology.

Configurable Semantics. All traceability links are stored in an explicit traceability model (G6). When there is an implicit connection between models (e.g., from one UML component to another), YT provides a rule-based language that can be used to specify how explicit links can be automatically derived. If feasible, such derived links can be stored in memory and not persisted to simplify maintenance.

YT was designed to be a highly configurable tool. Consequently, it must be configured according to the needs of a client. A technical project leader from the company stated that default configurations are not provided as needs differ substantially between companies and even between projects in the same

[9] `http://www.yakindu.de/traceability/`

company. Traceability models can be configured for each customer, with a few restrictions concerning the general shape of a link (e.g., a "link" can connect only two things). Another obvious restriction is that all model types to be connected must be supported by corresponding tool adapters. If required, new tool adapters can of course be developed for a customer who is ready to pay for it. YT thus adheres to (G7) with a few constraints.

Consistency Specification. YT does not dictate which approach should be used for traceability maintenance. According to the technical project leader, the tool can be used to support several processes in combination with other tools (G8). In combination with a VCS, for example, YT can enforce access restrictions for editing the traceability model.

Both manual and automatic creation of traceability links is possible (G9). Automatic links are defined by a rule-based language as opposed to employing machine-learning techniques, indicating that (G10) is desired by clients.

3.5 Related Work

Most of the research on traceability maintenance is in the area of (semi-)automatic maintenance of traceability links. These approaches can be categorised as Transformation-Driven, Event-Driven and Rule-Based.

Transformation-Based Approaches take advantage of the fact that (model) transformations can be suitably enriched to additionally produce traceability links. In general, traceability maintenance in this context requires *incremental* transformation approaches, for which the case when both source and target models evolve separately is challenging [59, 60]. This approach also assumes that all artefacts are created via model transformations, which is often not the case in practice (yet). An example for this approach has been proposed by Fockel et al. [185], who describe an approach for semi-automatic establishment and maintenance of traceability links in the automotive domain. Another example is [195], where the authors use a graph-transformation based approach to define, identify, and maintain traceability links.

Event-Driven Approaches leverage events occurring during software development activities to maintain traceability links. As a simple example, deletion of an artefact can be used as a trigger to delete all traceability links connected to it. Research employing this techniques include [196], where a publish and subscribe mechanism is used to connect traceability maintenance tasks to certain events.

Rule-Based Approaches use rules to determine when traceability links should be generated. For example, Spanoudakis et al. [23] define rules based on attributes of artefacts, for creating traceability links between requirements, use cases, and analysis object models. Traceability links are maintained by re-evaluating the rules. Rule-based approaches can be combined with event-driven approaches such as in [29, 168], where traceability maintenance is conducted in two phases: recognising changes based on events, and (re-)evaluating the rules governing link updates.

Even though some of the factors discussed in this paper have been mentioned in the literature (e.g., as requirements in [197]), we are not aware of any categorisation of primary factors together with guidelines for traceability main-

tenance such as we provide. In an experience report, Kirova et al. [32] propose technology recommendations for a traceability tool. These recommendations support our proposed guidelines especially on versioning and configurable semantics. However, their research was not focused on traceability maintenance and is based on data from only one company. The research by Gotel and Mäder [16] provides guidelines for selecting a TM tool. Their guidelines are directed at end users, while ours are aimed more at developers of such TM tools.

3.6 Threats to Validity

With regards to data source (S1), the elicited traceability requirements were from industrial and academic partners in the *automotive* industry. It is thus questionable to what extent the requirements can be generalised to other domains.

Data source (S2) was part of a larger traceability study whose main focus was on TM in general. This broader scope could have influenced the interview parts related to traceability maintenance. To minimise this threat, we complemented the data with an analysis of existing TM tools and interviews with TM tool developers and expert users (Section 3.4).

A further threat to the validity of our study is that nine of the interviews from S2 were conducted in German and then translated as accurately as possible to English. For consistency and readability, all interview quotes were also rephrased using our established terminology in the paper (e.g., model instead of artefact, document, or file).

Lastly, researchers' bias in identification of the factors could have affected the results. We mitigated this threat by involving four researchers during analysis and discussing the results with TM tool developers and expert users.

3.7 Conclusion and Future Work

In this paper, we presented factors that greatly influence to what extent a TM solution can support viable traceability maintenance. We suggested guidelines for each factor, which should be followed to avoid potentially negative consequences of certain (combinations of) design decisions.

To evaluate our guidelines, we analysed existing commercial and fairly established or at least reasonably successful TM tools. Our results show that while our guidelines are mostly adhered to (indicating that this is necessary for successful traceability maintenance), configurability and the level of automation can be improved.

Our results and conclusions are backed by interviews with our project partners, a broad range of software development stakeholders, and expert TM tool users and developers.

Our findings can be used by practitioners to develop and select TM tools. Researchers can build up on our findings to create more applicable (automated) TM methods and techniques that take practitioners' needs into consideration.

As future work, we plan to continue ongoing development on an open

source TM tool Capra,[10] which will be used and evaluated in the context of the Amalthea4public project. To cater for the wide range of project partners, we aim to address especially configurability (G7) and integrating manual and automation techniques (G9) better than existing TM solutions, applying model-driven technologies to enable truly domain-specific traceability solutions.

[10] http://salome-maro.github.io/TraceabilityManagement

Chapter 4

Paper C

Capra: A Configurable and Extendable Traceability Management Tool

S. Maro, J.-P. Steghöfer

24th International Conference on Requirements Engineering (RE2016), Beijing, China, September 12 - 16, 2016.

Abstract

Traceability is a known problem both in academia and industry. One of the main challenges is that there is no one solution that will solve traceability problems for everyone in industry. Traceability needs are dependent on the context of the organization and can differ from project to project in the same organization. To cater for this problem we have developed Capra, an open source, flexible, configurable and extendable traceability management tool. Capra can be tailored according to specific traceability needs of individual projects and organizations.

4.1 Introduction

Traceability in software development refers to the ability to link software artifacts like requirements, code, and tests throughout the development life cycle [5]. Traceability facilitates impact analysis, verifying that requirements have been implemented and tested and in some domains, e.g. in the automotive domain, it is required for fulfillment of safety standards such as ISO 26262 [75]. A major challenge for traceability tool developers is that traceability needs differ from company to company and even from project to project [32, 108]. To build a tool that fits a certain company, one needs to analyze the needs of that company and in most cases the solution will be feasible for that company only. This is not a good business model for commercial tool vendors or open source tool developers who want the same tool to be used in multiple companies. To solve this, a traceability tool needs to be configurable and extendable in such a way that it can be customized specifically to fit the needs of various companies. This is also referred to as *traceability fit for purpose* [85].

We collected requirements for a traceability tool that can integrate into a workflow for the development of embedded systems from a number of industrial partners, mostly in the automotive domain. Based on the collected requirements, Capra[1] has been developed. The choice of which parts of the tool should be configurable is based on the variations that we encountered in the requirements from the different companies. The requirements with the highest priority are the following:

[a] As a user, I want to create traceability links to arbitrary artifacts.
[b] As a project manager, I want to define custom traceability link types for projects.
[c] As a user, I want to visualize artifacts connected by traceability links through a matrix or graph view.

Our current implementation of Capra supports all these requirements by allowing the end user to create, update, and visualize traceability links. It also allows defining custom link types and extending the tool to support arbitrary artifacts.

In comparison to existing tools, Capra supports traceability between arbitrary artifacts (as compared to, e.g., DOORS[2] that, at least off the shelf, only supports traceability between requirements) and a higher degree of customisability (as compared to other tools such as ReqCycle[3] that does not allow modifying storage of traceability links and extending the targets of traceability links easily).

This extended abstract describes the architecture of the tool and an implementation of its default configuration.

4.2 Architectural Design

Capra is an Eclipse plugin and uses the Eclipse Modelling Framework (EMF) as its base technology. It stores the traceability model as an EMF model.

[1] http://salome-maro.github.io/TraceabilityManagement/
[2] `http://www-03.ibm.com/software/products/en/ratidoor`
[3] `http://www.polarsys.org/projects/polarsys.reqcycle`

The tool relies on the Eclipse Extension mechanism[4] and provides extension points for parts of the tool that can be customized. Based on requirements we collected from our project partners, the tool is customisable at three points: i) the types of links to be supported; ii) which types of artifacts can be traced to; and iii) how the links should be stored. Figure 4.1 depicts the extension points and the rationale for each of them is described in the following.

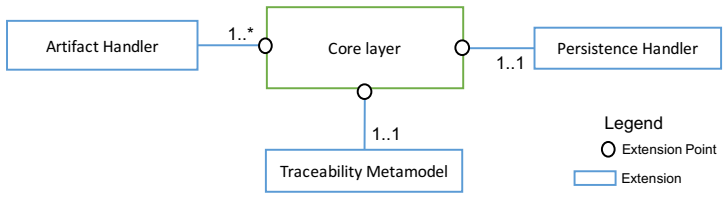


Figure 4.1: Extension points in Capra

4.2.1 Traceability Link Types

Depending on the company, development context, and process used, the traceability links required can differ [85, 198]. For example, traceability links for a company developing web-based solutions are not the same as links for companies developing embedded software. To address the different link types, the tool offers an extension point for the traceability metamodel (see Figure 4.1). Here the end user (company), can define the types of links through a metamodel and supply it to the tool. Examples of link types are "verifies", "implements", "refines", "related to" etc.

4.2.2 Supported Artifact Types

Software development usually involves a number of activities such as requirements engineering, design, implementation and testing. In most cases, each of these activities use different tools and produce artifacts of different formats. A traceability tool needs to ensure that the different formats can be traced to and from. Since different companies use different tools, it is not easy to foresee which formats a traceability tool should support. This problem of diverse artifacts existing in the development environment has been noted by several studies on traceability [5, 182]. Our tool offers an extension point for Artifact Handlers which allows adding artifact formats based on the needs of the end users.

As discussed, Capra stores the traceability links as an EMF model. To be able to support tracing to other formats, EMF representations of these other formats are required. Implementing an extension for a certain format means providing an EMF representation of that format to the tool using the artifact handler extension point.

[4] `https://wiki.eclipse.org/FAQ_What_are_extensions_and_extension_points`

4.2.3 Persistence Extension Point

The storage of traceability links is another factor that can vary depending on company policies or project set-ups. For some cases it makes sense that there is a traceability model per project while in some cases there can be one traceability model for the whole workspace. The extension point Persistence Handler allows defining such storage locations. It will also allow integrating the traceability model with versioning solutions such as EMF Store, CDO or Git.

4.3 Functionalities of Capra — The Default

One of the challenges of creating a configurable tool is that it cannot be used out of the box without a considerable effort going into the configuration first. Since Capra is also very flexible, its core is not usable without any extensions provided to the extension points. To deal with this, we have implemented a default configuration that offers basic extensions to the tool.

Currently, the default configuration of the prototype offers a simple traceability metamodel that supports creation of traceability links that have source and target of any supported artifact type. As shown in Figure 4.2, there are six supported artifact types: Java code (up to method level), C/C++ code (up to function level), files (such as PDF or MS Word), task tickets supported by Mylyn, and test case execution from a continuous integration tool such as Hudson. For storage of the traceability links, the prototype implements an extension to the persistence extension point that stores all links created in the same work space in one folder.

The tool also has functionality for visualization of the traceability links. The links can be visualized in a matrix as well as a graphical format with artifacts represented as nodes and links represented as edges. Figure 4.3 shows such a graphical view extracted from a Heating Ventilation and Air Conditioning (HVAC) system. The example shows a requirement specification about a "blower" connected to a feature represented by a class. The class is connected to a component and a PDF file, and lastly the component is connected to a state machine and another component.

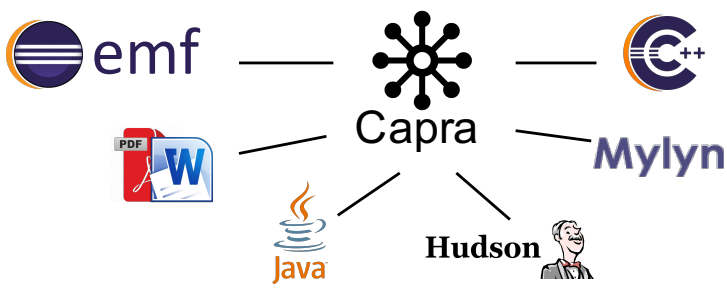


Figure 4.2: Artifact types currently supported by Capra

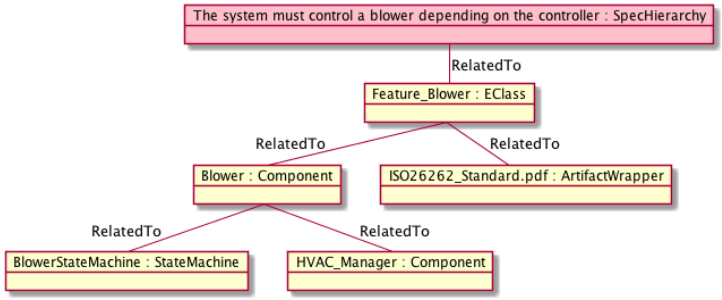


Figure 4.3: Graphical representation of artifacts connected by traceability links

4.4 Conclusions and Future Work

The main contribution of Capra is to provide a configurable and extendable open-source traceability solution. In order to build a flexible tool, one needs to design for flexibility from the start. For future work we aim to incorporate features such as versioning to support trace link maintenance and collaboration features such as discussion, chats and voting in the context of a traceability link in order to improve trace link quality.

Acknowledgment

This work is part of the AMALTHEA4Public project funded by the ITEA Eureka Cluster programme.

Chapter 5

Paper D

Vetting automatically generated trace links: What information is useful to human analysts?

S. Maro, J.-P. Steghöfer, J. Hayes, J. Cleland-Huang, M. Staron

26th International Requirements Engineering Conference (RE) (pp. 52-63). IEEE.

Abstract

Automated traceability has been investigated for over a decade with promising results. However, a human analyst is needed to vet the generated trace links to ensure their quality. The process of vetting trace links is not trivial and while previous studies have analyzed the performance of the human analyst, they have not focused on the analyst's information needs. The aim of this study is to investigate what context information the human analyst needs. We used design science research, in which we conducted interviews with ten practitioners in the traceability area to understand the information needed by human analysts. We then compared the information collected from the interviews with existing literature. We created a prototype tool that presents this information to the human analyst. To further understand the role of context information, we conducted a controlled experiment with 33 participants. Our interviews reveal that human analysts need information from three different sources: 1) from the artifacts connected by the link, 2) from the traceability information model, and 3) from the tracing algorithm. The experiment results show that the content of the connected artifacts is more useful to the analyst than the contextual information of the artifacts.

5.1 Introduction

Traceability is regarded as important in software and systems engineering [14]; however, its adoption across many industrial sectors is still quite low [34]. One of the main inhibitors to adoption is the cost of creating and maintaining traceability links [30, 199]. To combat this challenge, many automated techniques for creating and maintaining trace links have been proposed [200] including techniques based on information retrieval [20, 25], machine learning [21], deep learning [201], rule-based [23], and repository mining [202]. Automated techniques are promising, since they could potentially eliminate the manual work of creating and updating trace links. However, current solutions do not yield perfect results in terms of either precision or recall [25, 203]. Human analysts therefore need to inspect the generated trace links and make a final decision on their correctness. We refer to the task of inspecting automatically generated links to confirm true links and reject false links as "vetting." The task of vetting trace links is tedious, and it has been observed that, in some cases, instead of improving the accuracy of the generated trace links, human analysts actually decrease their quality [36].

Researchers have conducted two types of studies on the traceability vetting process: studies that investigate the impact of varying the accuracy of the generated traceability model and studies that build and evaluate tools to support analysts. Regarding the former line of research, Cuddeback et al. [36] showed that while human analysts tend to improve the trace model if it has a low precision and recall, they tend to degrade the quality of the trace model if it initially has high precision and recall. Regarding the latter line of research, Hayes et al. [45], e.g., developed RETRO, a traceability management tool that generates links automatically and presents them to a human analyst for vetting. The tool offers features such as global tracing, local tracing, and filtering the generated links based on a score from the tracing algorithm. The main focus of such studies is the usability of the tool and which benefits for vetting trace links it provides over a manual tracing approach. However, further empirical studies are needed to understand which factors influence human analysts' decisions when vetting trace links in order to improve this process [204, 205] within specific software engineering contexts [206].

In this study, we take a different perspective and hypothesize that the task of vetting automatically generated links can be improved if the traceability tool provides useful information to the human analysts. We specifically investigate how *context information* can be useful to the human analyst. In our study, we use the definition of context information from Abowd et al. [79]: context information refers to any information that can be used to characterize the development artifact – e.g., the meta data of an artifact, such as the date it was created or modified. In contrast, the content of the artifacts, such as the code in a Java file or the textual description of a requirement, is not considered context information. Our assumption is that offering context information together with the content of the artifacts will improve the decisions made by human analysts.

Our study has two main contributions. First, we conduct an empirical investigation investigating what context information is needed by human analysts. We collect data from practitioners and compare it to existing literature. Second, we investigate the effect of context information for supporting the

human analyst during the vetting task. The study addresses the following research questions:

RQ 1: What context information is useful to human analysts when vetting trace links?

RQ 2: To what extent does this information help analysts to make correct decisions?

The remainder of the paper is structured as follows. Section 5.2 provides an overview of the related work in the area of vetting trace links while Section 5.3 describes our research methodology. Our results are described in Sections 5.4, 5.5, 5.6, and 5.7, while Section 7.6 discusses the results with respect to existing research. We conclude with a discussion of threats to validity in Section 5.9 and then summarize our findings in Section 5.10.

5.2 Related Work

Related work primarily falls under the two areas of vetting trace links and tools for supporting human analysts in the vetting task. We discuss each of these areas in this section.

5.2.1 Performance of human analysts in vetting trace links

Several studies confirm that human analysts make mistakes when vetting automatically generated trace links. Cuddeback et al. [36] studied the performance of human analysts in evaluating the correctness of generated trace links. They showed that humans often degrade the accuracy of generated links by accepting wrong links and rejecting correct links. As a result of these findings, researchers have studied factors that contribute to human analysts' decision making in order to design tools to improve the quality of human-vetted links.

Dekhtyar et al. [42] investigated several factors that may influence the accuracy of trace links produced as a result of human decision making. They showed that the quality of the initial trace link set and the effort in analyzing the links both made a difference in the final accuracy. Sets of trace links that started with high recall and precision tended to decrease in both recall and precision, while sets that started with low recall and low precision led to improvements in both recall and precision. Similarly, a starting set with a high precision and low recall led to improvements in recall but decreases in precision, and vice versa. They also observed that analysts who reported investing more effort in vetting the links often ended up reducing recall by rejecting true links. These findings align with those from similar experiments reported by Cuddeback et al. [36] and Kong et al. [44]. One possible explanation for their results is the presence of gray links, i.e., those links which capture meaningful associations between artifacts, but are ambiguous because they are only relevant for certain software engineering tasks [206].

Additionally, Kong et al. [207], studied how humans make correct and incorrect decisions by investigating logs recorded during the vetting process.

They identified several strategies that analysts use to make decisions such as "first good link" (participants focusing on finding the first good link) and "accept-focused" (where participants only accepted links and never explicitly rejected links). This study also showed how the tracing experience of the analyst and effort spent on vetting traces affects the quality of the final set of traceability links.

Another study reported by Dekhtyar and Hilton [208] investigated the strength and weaknesses of human analysis versus automated tracing techniques in order to identify how the two approaches could best complement each other. They proposed using an automated technique that identifies links which human analysts would be more likely to miss.

Our study, however, takes a different approach by studying context information that can be presented to the human analyst in order to make the decision process easier and more accurate. Kong et al. [44] already observed that during the vetting task some analysts go back to review the connected artifacts and other artifacts in the data set before they decide on a link. This implies that the analysts seek to understand the artifacts and how they fit together in the data set in order to make decisions.

5.2.2 Tools for vetting trace links

There are several research tools that support the vetting of trace links. It is important to note that not all tools that can generate trace links offer functionality for vetting those links. Many research tools stop at the generation stage (see, e.g., [167]), because they are focused on presenting or improving certain automated techniques to generate the links, and are not interested in further steps such as vetting or utilizing links. The main functionalities provided by tools that support link vetting are: generating links, presenting them to the analyst, and finally allowing the analyst to accept correct links and to reject incorrect links. Some tools also allow the analyst to search for missing links [45,209] and to perform coverage analysis for completeness of links [47]. To the best of our knowledge, there are six tools that support vetting of trace links: RETRO [45], ADAMS [210], Poirot [46], TraCter [211], TraceME [47], and ART-Assist [212]. A deeper analysis of these tools is presented in Section 5.5.

5.3 Research Method

The study was conducted using the design science research method [213, 214] in which a problem is iteratively investigated while implementing suitable artifacts to solve the problem and evaluating the effectiveness of the solution. We utilized a combination of techniques based on interviews, literature review, and a series of controlled experiments. The process was conducted in three distinct iterations.

5.3.1 Interviews and Identification of Existing Tools

In the first iteration, we conducted a series of semi-structured interviews to investigate what information affects human analysts' decisions when vetting

Table 5.1: Interview Subjects

Subject	Role	Domain
A	Business Analyst	Finance
B	Software Architect	Automotive
C	Practice area lead, software and license handling	Telecommunication
D	Software Lead	Military
E	Requirements engineering researcher	IT Consultancy
F	Research Fellow	Automotive
G	System Engineer	Software Development
H	Requirements engineering researcher	IT Consultancy
I	Verification and Validation analyst	Space
J	Senior Software Architect	Automotive

traceability links. We first designed an interview guide[1], for which the questions were reviewed and re-written by four researchers across several iterations. We then tested this interview guide by conducting one pilot interview in order to establish its soundness and to improve the guide. After the pilot, we interviewed ten practitioners with experience in creating and maintaining traceability links. We used convenience sampling where our aim was to interview practitioners with experience in traceability while maximizing diversity in our interview sample. All the interviewees were recruited through our personal connections and were from different domains. Table 5.1 provides a summary of our interviewees, their roles, and the domains in which they work. The interviews focused on identifying which types of information analysts used to create traceability links and how trace links are evaluated in the interviewee's company. We assume that this same information should also be available during the process of vetting automatically generated trace links. In addition, we asked the interviewees about their experience with automatic tools for traceability link generation and how they would expect such tools to work. For each interview, we summarized a set of information and information sources discussed by the interviewee in a spreadsheet.

To further strengthen our results, in the second iteration of our study, we researched existing literature on tools that support the vetting of trace links that have already been published in scientific literature. Our aim was to identify if these tools already offer the information collected from the interviews and to identify any gaps that might exist. We conducted our literature search by starting with previously identified papers on trace link vetting and then using snowballing to extend the scope. Many of these papers are already discussed in Section **??** describing related work.

[1] https://tinyurl.com/yauypdhn

5.3.2 Experimental setup

To validate the usefulness of context information during the vetting process, we created a prototype by extending an open source traceability management tool, Eclipse Capra [69]. We chose to extend Eclipse Capra for this study because it is a customizable open source tool that contains basic traceability features such as creating and visualizing trace links. Additionally, two of the authors are developers of the tool. Details of how Eclipse Capra was extended are provided in Section 5.6. Using the prototype we conducted an experiment with 33 participants in order to understand how various types of context information affects the human analyst's decision making process.

Experiment Variables: The experiment was designed to investigate whether providing analysts with contextual information from connected artifacts would affect their performance in vetting trace links. Examples of contextual data include meta data (e.g., date created), attributes (e.g., status or priority), location (e.g., subsystem), and connectivity to other artifacts. The independent variable was therefore context information of the artifacts, where one group was given contextual information and the other was not. The choice of contextual information to include in our study was driven by results from our interviews and literature review as reported in Section 5.4. We measured three dependent variables: 1) recall of the final trace links; 2) precision of the final trace links; and finally 3) the number of links investigated by the analyst within the allotted time.

To reduce the number of confounding factors, we controlled three variables: 1) the initial precision of the trace links, 2) the initial recall of the trace links, and 3) the order in which the experiment subjects vetted the trace links. All analysts were instructed to review the provided list of links from top to bottom as provided to ensure that they inspected the same links without intentionally skipping any links. This also made it possible for us to identify links investigated by the analysts for which they did not indicate a decision.

In the experiment, an analyst who produced trace links with higher final recall, higher final precision, and who investigated more links was considered to have outperformed an analyst with lower final recall, lower final precision, and a lower number of investigated links.

Based on our research questions, we formulated the following hypotheses to investigate the three dependent variables:

H_{o1} Providing human analysts with context information about the artifacts connected by trace links has no effect on the precision of the final set of trace links

H_{o2} Providing human analysts with context information about the artifacts connected by trace links has no effect on the recall of the final set of trace links

H_{o3} Providing human analysts with context information about the artifacts connected by trace links has no effect on the number of links investigated during a given time period

These hypotheses guided our evaluation into how and to what extent context information is useful to the analyst (RQ2).

5.3.3 Experiment Materials

We describe the experiment artifacts that the candidates interacted with as well as the data collection instruments.

Experiment Artifacts: We used Medfleet, a system developed by Software Engineering graduate students as part of a five month studio course [215]. We selected this system because it contains realistic artifacts of a software development project, and has been used in a previous publication [216] with a set of verified traceability links. Medfleet enables its users to request emergency medical kits to be delivered using small Unmanned Aerial Systems. The project artifacts include requirements, environmental assumptions, architectural components, code, and fault descriptions. Requirements, assumptions, and faults were originally captured in Jira, while code was written in both Java and Python and stored in GitHub. The trace links provided by the project served as a gold standard and consisted of 288 true links. In the experiment we used a subset of the artifacts – requirements, Java code implementing the mission control subsystem, assumptions, and faults to reduce the scope and make the experiment more manageable. We used the Vector Space Model with Term Frequency-Inverse Document Frequency (TF-IDF) technique to generate three types of traceability links: links from requirements to code, links from requirements to assumptions, and links from requirements to faults. TF-IDF is an information retrieval technique that not only uses text similarity to predict how similar two artifacts are, but also uses term frequency [217]. The weight of the terms is calculated as a product of the frequency of the term in a given document and the inverse of the frequency of the term in all the documents. This technique gives an indication of how important a term is in a given document. The total number of links generated was 1239. The precision and recall for requirements to code links was 4.2% and 33% respectively, for requirements to assumptions links was 8.6% and 56% respectively and for requirements to faults links was 7.7% and 54% respectively.

Data Collection Instruments: We created two questionnaires to collect data from our experiment: 1) a pre-experiment questionnaire that collected information about the participants' experience with software development, use of the Eclipse IDE (on which Eclipse Capra is based), and experience with traceability, and 2) a post-experiment questionnaire that collected feedback on the experiment and the different features of the tool with which users interacted. Additionally, we recorded the screen during the experiment. To collect information about the vetted links, we stored all the links that the analyst accepted in a list of accepted links and all the links that the analyst rejected in a list of rejected links. These lists were stored as EMF models in Eclipse Capra.

Experiment Subjects: We used convenience sampling to identify diverse participants from our personal connections. As a result, the experiment was conducted with 33 participants of which six were Bachelor students, eight were Masters students, twelve were PhD students, and seven were industry practitioners. The students were all software engineering students from two universities and therefore had all taken several courses on software development. The subjects were randomly divided into two groups, the control group (16 subjects) and the experiment group (17 subjects). Out of the 33 subjects

who took part in the experiment, we discarded the results of five subjects because they did not follow the instructions of the experiment, for example, by evaluating trace links in a different order from the instructions or evaluating only one type of trace links. These results were excluded in order to avoid bias.

Experiment Groups: The control group was provided with a version of the tool which did not display context information, while the experiment group was provided with a version of the tool with context information of the connected artifacts. This means that the control group had features F1 to F8 and the experiment group had features F1 to F8 and additionally features F9 and F10 (cf. Section 5.6).

Experiment Procedure: All experimental sessions began with one of the researchers giving a scripted brief introduction to traceability and automated techniques of generating trace links. This was followed by explaining the Medfleet system and the vetting task to the participants. During this session, participants were allowed to ask questions. The instructions of the experiment were also distributed in paper format for participants to read [2]. Before the experiment, participants filled in the pre-experiment questionnaire. The participants were then given 45 minutes to vet as many links as they could. At the end of the experiment, we collected the final traceability models the participants produced as well as video recordings of the screen for the entire 45 minute vetting session. The participants also filled in a post-experiment questionnaire containing questions about the task and which features of the tool they found useful. The questionnaires for each group are available online [3][4]

5.4 Interview Results

From the interviews, we learned that the task of vetting trace links is conducted in companies, even when trace links are created manually. In safety-critical domains, trace links must be carefully reviewed before submission to the certification body [218] . The information used to evaluate the correctness of a trace link is therefore used during both the link creation and the link assessment processes. As a result, information collected from the interviews was derived from two activities, that of creating and reviewing the links. We categorized the information that the interviewees reported into three main categories of information derived from 1) connected artifacts (Section 5.4.1), 2) the traceability information model showing connections between artifact types (Section 5.4.2), and 3) results from the tracing algorithm (Section 5.4.3). Additionally, interviewees reported how they would like this type of information to be represented or displayed (Section 5.4.4). From these results we ultimately selected specific contextual and content-based elements to be included in our tool and used in the evaluation.

5.4.1 Information from the connected artifacts

Six out of ten interviewees reported that they create trace links based on their knowledge of the system. They use their experience to determine if two artifacts

[2] https://tinyurl.com/y98jpdtg
[3] `https://goo.gl/forms/zdY23Gqjk1rixF4U2`
[4] `https://goo.gl/forms/jMRKW9yWitHDj4JI3`

(e.g., a specific requirement and a specific Java class) should be connected or not. However, when asked what information they would need if they did not have such system experience, they reported the following:

- The content of the connected artifacts: this refers to the information that makes up the artifacts, e.g., the content of the Java file represented by the actual lines of code, or content of a requirement represented by its textual description.
- The meta data of the connected artifacts, such as who created it, when it was modified, and who modified it.
- Other artifacts connected to the artifacts: This refers to other development artifacts that already have established links with the investigated artifacts, e.g., when deciding if requirement X is connected to a Java file Y, one may first want to know which other requirements are related to requirement X or to the Java file Y.
- The location of the artifacts in a project, system, or subsystem.

5.4.2 Information from the traceability information model

Before creating or reviewing trace links, all the interviewees reported that it is necessary to understand how the different artifacts in the project are related to each other. For instance, interviewee E reported that the company has a metamodel that specifies how the artifacts should be connected. Such a metamodel or traceability information model (TIM) contains information indicating specifically which types of artifacts may be linked together within a given project. For example, a link might be allowed from an acceptance test to a requirement, but not directly from the acceptance test to code. In summary, users need the following information from the TIM:

- A definition of which links are allowed and which are disallowed. The TIM may also contain additional information about cardinality constraints.
- The type of link that is being created or reviewed. For TIMs that contain diverse link types (e.g., tests, refines, describes), the analysts need to understand the type of link they are currently vetting before making a decision.

5.4.3 Information from the tracing algorithm

While many interviewees were aware of techniques for automatically generating trace links, only three had actually used such tools and had first-hand experience of tracing algorithms. For interviewees that had no experience with these tools, the interviewer carefully described how they worked. We then asked all participants what information they would expect to see if they used an automated tool to create links. This was first asked as an open question, and if the interviewees did not have any ideas, we suggested options. All the interviewees agreed that a confidence score for each link would be useful.

Additionally, some of the interviewees agreed that for information retrieval techniques that use text matching, seeing which exact words or phrases from the source artifacts matched words or phrases in the target artifact would be beneficial. Therefore we concluded that the following information should be available for evaluating trace links:

- A score representing the similarity of two artifacts. This score is calculated by the information retrieval algorithm that is used to generate the links.
- Words or phrases from the source artifact that matched words or phrases in the target artifact.

5.4.4 Presentation of the information

Especially for large projects, the number of artifacts and trace links can be overwhelming and therefore tool support is critical for creating and reviewing links. Our interviewees reported that when reviewing trace links, they would like to search for artifacts and filter the links that they are reviewing. Additionally, it was reported that being able to review one type of trace link (e.g., requirements to code) is beneficial, compared to viewing all the trace link types at once.

Two out of ten interviewees reported that they would like to have a graphical representation of the trace links. However, they noted that since a large amount of trace links can lead to large graphs which are complex to read, the traceability management tool should allow filtering of the links to display manageable graphs.

From this category, we identified two features that should be included:

- Ability to search for and filter trace links.
- Ability to view trace links in a graphical representation.

5.4.5 Context information

Since the interviewees reported generally on what information is useful when vetting links, we collected this information and used our definition of context information (information that can be used to characterize the development artifact) to derive context information relevant for trace links vetting. This information is summarized and exemplified in Table 5.2.

5.5 Investigation of existing tools

We compared the results from the interviews to available literature on vetting traceability links. Specifically, we investigated research tools that support vetting of traceability links. Our literature search started with a few papers that are prominent in the field, e.g., [45], [36], [42] and used snowballing to acquire more papers. We found six tools described in scientific literature that support traceability link generation and the activity of vetting traceability links. These tools and their features are summarized in Table 5.5 along with the extended Eclipse Capra. Our investigation was focused on features that provide information to the human analyst and did not focus on analyzing the

Table 5.2: Context Information

Context Information	**Description**	**Example** (w.r.t. Requirement (RQ-01))
Meta data	This refers to data describing the artifact e.g., who created the artifact, when it was created, when it was modified	created on: May 19, 2016, createdBy: SMaro
Location	Where the artifact is located and in which system	MedFleet/Requirements.xlxs
Connected artifacts	Other artifacts linked to the artifact in question	Assumption (A-01), Fault (F-01)

different tracing strategies provided by the tools. We used this knowledge from the existing tools as inspiration for our own implementation of features that were suggested by the interviewees. The implementation decisions and details are discussed in the next section.

Information	RETRO	Poirot	TraceME	AdamsTrace	TraCter	ART-Assist	Eclipse Capra
Navigation to full artifact content	Yes	No	Unk.	Unk.	Yes	Yes	Yes
Score from the tracing algorithm (Similarity measure)	Yes	Yes	Yes	Yes	No	No	Yes
Matching terms from the source in the target artifact	Yes	Yes	No	Yes	No	No	Yes
Text search	Yes	Yes	Unk.	Yes	Unk.	Unk.	No
Trace link type	N/A	No	Yes	Yes	Yes	Yes	Yes
Graphical representations	No	Yes	Yes	No	No	No	Yes
Accepted links	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Location of the artifact	Unk.	No	Unk.	Unk.	Yes	Yes	Yes
Summary of the artifacts (e.g., Java class documentation)	No	No	No	No	No	Yes	No

Table 5.3: Information provided by existing tools that support trace links vetting. “Yes": information/feature present; “No": information/feature not present, “N/A": not applicable; “Unk.": not enough information available to determine if feature present.

5.6 Prototype implementation

In order to perform our evaluation using a controlled experiment, our prototype needed to automatically generate trace links given a set of development artifacts and then to present these links for vetting, together with context information, to the human analyst. To generate the initial set of trace links, we extended Eclipse Capra to use the Vector Space Model – Term FrequencyâĂŞInverse Document Frequency (TF-IDF) [217]. This technique is proven useful in previous studies [212].

To provide the ability to vet the generated traceability links, we implemented ten features identified through the interviews. Features 1 to 8 provided basic functionality for vetting links, while features 9 and 10 presented context information. Our choice of what context information to include and evaluate is based on the results of the interviews. Three of our interviewees reported that meta data of the connected artifact was useful, four mentioned location of artifacts (e.g., artifact X and Y are both from the same subsystem or package) and five mentioned relationship to other artifacts. Because our aim was to study whether a type of information, and not its exact representation, was important, we made every effort to select a simple solution for each type of information. We provide the rationale for the way each feature was implemented below and illustrate some of these features in Figure 5.1.

F1: A list of trace links generated from the tool: We displayed trace links in the form of a tree list where the parent of the tree represents the source artifact and the children of the tree represent the target artifact. This type of display is what is known as local tracing, where the user can view links related to one artifact at a time. We chose this implementation since it was preferred by users in a previous study reported by Hayes et al. [45].

F2: Ability to open the artifacts connected by the links: To make sure that users can access the content of the connected artifacts, we implemented functionality to open the connected artifact for each of the artifacts This feature is present in some of the tools, e.g., in RETRO, TraCter, and ART-Assist. In RETRO, the content of the requirements are displayed as text. In TraCter and ART-Assist, connected Java files are opened in a pop-up window. In our case, we decided to use the native environments of the tools that were used to create the artifacts. For instance, all Java files are opened in a Java editor so that features such as syntax highlighting are available.

F3: A display of similarity scores generated by the tracing algorithms: With the exception of TraCter and ART-Assist, the rest of the tools display the similarity score from the trace generation algorithm. This feature has also been highly ranked by user studies performed on RETRO [45] and ADAMS [20]. There were two possibilities to display this score. One way is to display the raw value (or a percentage) and the other way is to translate this into a confidence value as done in Poirot [46]. We chose to display the values as raw numbers since this has been shown to work and was highly rated [20, 45] while there are no user studies on the alternative by Poirot.

F4: A graphical representation of the links: We implemented a graphical representation of the links that displays trace links as a graph. The source of the trace link was the root node of the graph, while the targets of the trace link were the child nodes. We did not implement any transitive links, i.e., displaying more than one level of traceability, because three of our interviewees had indicated that for the purpose of vetting the trace links, they could only deal with one level of trace links at a time.

F5: A view of the Traceability Information Model (TIM): This feature was requested by all interviewees. Additionally, since we wanted our tool to support diverse link types, displaying the TIM to the user gives information on which links are allowed and the different constraints on the different link types. Several tools such as RETRO, only support one type of trace link, i.e., tracing high level requirements to low level requirements, and therefore displaying the TIM was not a required feature.

F6: Ability to see the link type: This feature is related to feature F5. When vetting the trace links, the user should be able to know which link type they are currently vetting. We implemented this by adding the link type name, e.g., "satisfies," "realizes," or "requirements to code" depending on how the TIM is defined for the project.

F7: Ability to see terms from the source artifacts that matched terms from the target artifacts: This feature enables the user to see which term in the source artifact matched terms in the target artifact. The most common way of implementing this is through highlighting these terms in the source artifact and in the target artifact [45, 46]. In our case, due to technical limitations (it is tricky to implement multiple highlighting in the Java editor),we followed a simpler solution and used markers to indicate lines where these terms occurred. Since there can be many terms, we only displayed the top three terms according to their TF-IDF weights. This feature was only implemented for Java files.

F8: Ability to accept a link and reject a link: This feature was implemented to allow the analyst to accept and reject links by using the trace link list provided. The analyst could accept/reject one target for a link at a time or could accept/reject all suggested targets at once. This kind of implementation has been shown to work in RETRO and TraceME. When an analyst accepts a link, it is removed from the candidate trace link list and added to a list of confirmed links. When an analyst rejects a trace link as incorrect, the link is removed from the candidate trace links and added to a separate list of rejected links. Storing both the list of accepted links and rejected links enables the tool to keep track of what the analyst has already vetted in order not to show these links to the analyst again for vetting.

F9: **Ability to hover over the connected items and see context information for the artifacts**: Displaying context information (e.g., date created, date modified, who created and modified the artifact, and the location of the file) was a new feature that has not been implemented in

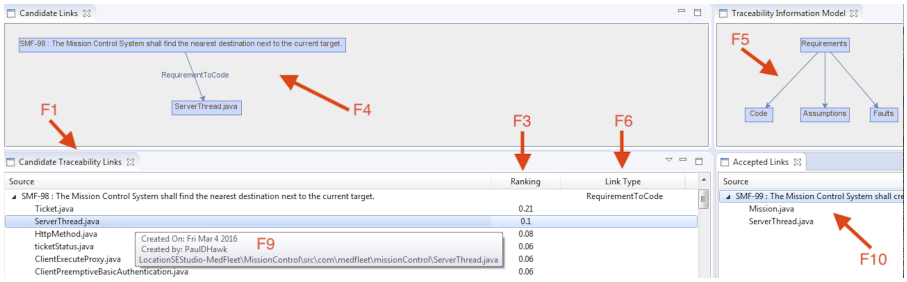


Figure 5.1: A screen shot of the trace links vetting tool with indicators for the different features.

	<6 months	**6-12 months**	**1-3 years**	**3-6 years**	**>6 years**
Experience with software development	2	2	11	5	8
Experience with Eclipse	9	2	9	3	5

Table 5.4: Software development experience of the experiment subjects

existing tools, with the exception of the location of the artifact in ART-Assist. We implemented the display of the context information based on the suggestion by the interviewees and show the context information in a tooltip when the mouse hovers over an artifact.

F10: The ability to see already accepted links: Other links that have already been accepted and contain the artifact that is currently being inspected are additional context information. When analyzing links related to requirement X, e.g., the analyst would like to know if other links to requirement X already exist. While the existing tools that allow the analyst to view accepted links show the entire list, in our case the user can see only accepted links related to the artifacts the user is currently inspecting. If the user is vetting links related to requirement X, then the user can only see accepted links that contain requirement X either as a source or a target. This implementation is due to the fact that, in our case, the accepted links are a form of context information while other tools show accepted links to indicate progress to the user.

5.7 Experiment Results

In this section, we report results for all participants who successfully completed the study (i.e., 14 in the control group, and 14 in the experiment group). The results of the pre-experiment questionnaire are shown in Table 5.7. Most of our participants had at least one year of experience with both software development and Eclipse. However, only eight of our participants had experience with traceability activities.

Table 5.5: Experiment Results

Test Group						**Control Group**					
Subject ID	Accepted	Rejected	Investigated	Precision (%)	Recall (%)	Subject ID	Accepted	Rejected	Investigated	Precision (%)	Recall (%)
TS01	100	33	149	13	87	CI01	54	81	135	19	56
TS02	79	85	164	14	61	CS01	49	44	98	20	67
TS03	27	38	66	33	60	CS02	12	2	19	25	100
TS04	60	13	95	20	92	CS03	111	49	160	10	61
TS05	58	39	128	19	92	CS04	17	67	84	47	53
TS06	49	128	177	14	39	CI02	46	92	138	20	50
TS07	130	207	337	13	63	CS05	47	84	134	21	56
TI01	73	61	134	16	67	CS06	103	56	159	16	87
TS08	14	57	71	29	27	CS07	42	0	42	24	100
TS09	37	20	57	32	80	CI03	73	36	109	19	82
TS10	40	55	95	28	69	CS08	210	113	323	11	92
TS11	98	1	99	16	100	CS09	17	19	36	29	56
TS12	82	26	108	16	76	CS10	47	7	54	32	100
TS13	270	53	323	9	96	CSI04	64	120	184	16	50
Average	**79.79**	**58.29**	**143.07**	**19.50**	**72.02**	**Average**	**63.71**	**55**	**119.64**	**22.04**	**72.23**

We recorded the total number of accepted links, total number of rejected links, number of correct accepts, number of correct rejects, and the total number of links investigated. We also computed the final recall and precision.

As can be seen from the results in Table 5.5, the performance of the two groups was quite similar. Since the data was not normally distributed, we used the Wilcoxon-Mann-Whitney U Test [219] with an alpha of 0.05 to determine if there was a statistically significant difference between the two groups for the following three variables: 1) the recall of the final trace link set, 2) the precision of the final trace link set, and 3) the number of links investigated by the analyst in the given time. The results show that for all three variables, the difference is not statistically different between the two groups. Therefore we cannot reject the null hypotheses (cf. Section **??**).

To get a better understanding of why there was no significant difference between the two groups, we further analyzed the post-experiment questionnaire results. Each group had a post-experiment survey that asked about the different information provided in the tool and their perception of how useful this information was for the vetting task. Since the experiment group had the contextual information, we analyzed their perception on this extra information. The survey included three questions related to the extra information given to the test group. The first two questions were on the metadata of the connected artifacts. All the questions were to be answered on a 5-point Likert scale, where 1 is strongly disagree, 2 is disagree, 3 is neutral, 4 is agree, and 5 is strongly agree. The first statement was "Knowing who created the connected artifacts was useful when vetting the trace links." For this statement, eight of the respondents said they strongly disagree, three disagreed, only one responded with agree, and two did not answer the question. A similar trend was seen for the second statement which was "Knowing when the connected artifacts were created was useful when vetting the trace links." For this statement, eight respondents stated that they strongly disagree, three stated that they disagreed, while only two agreed to the statement, and one was neutral (cf. Figure 5.2). We were able to ask some participants why they thought this information was not useful during their vetting task. Most of them responded that since they did not know the system or the people involved in creating the system, this information was not useful. However, they noted that if they were involved in the development of the system and knew the participants, then this information might have been useful. This corresponds to what our interviewees reported, that their experience with the system is important when vetting trace links.

Figures 5.2 and 5.3 show that three features were perceived as most useful during the vetting task: 1) the ability to open the connected artifacts, 2) the ability to know which link type is being vetted, and 3) knowing the similarity score of the link from the algorithm. This information also explains why our null hypotheses could not be rejected since the content of the connected artifacts was considered more useful than contextual information of the connected artifacts.

We conducted further analysis on our data to understand if there was a significant difference between the performance of industry participants and the performance of students. Due to a lower sample size of industry participants (four in the control group and one in the test group), we could only compare this for the control group. For both the precision and total number of links

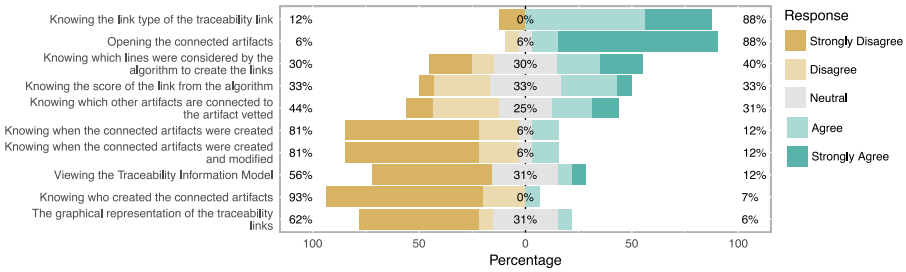


Figure 5.2: Results of the test group from the post-experiment questionnaire

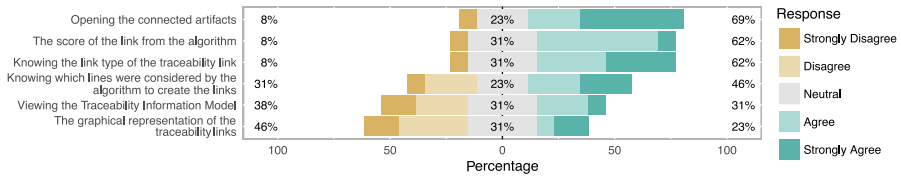


Figure 5.3: Results of the control group from the post-experiment questionnaire

vetted, using the Wilcoxon-Mann-Whitney U Test, there is no statistically significant difference between the performance of the two groups. However, for recall, we found that there is a significant difference between the groups, where the student group had slightly higher recall (see Table 5.6). Going back to the definition of recall (the number of correct links identified over the total number of correct links present in that set of links vetted), we can see that the students that had higher recall are those that vetted a small number of links. Subject F (in Table 5.6), e.g., did not reject any links. This therefore does not mean that the students had better performance, only that they vetted a smaller number of links and therefore reduced their chances of rejecting correct links.

Analyzing the post-experiment questionnaire for the industry participants and student participants, we see the same trend in the top three features that were found to be useful during the vetting process: 1) the ability to open the connected artifacts, 2) the ability to know which link type is being vetted, and 3) knowing the similarity score of the link from the algorithm (cf. Figure 5.4).

5.8 Discussion

In this section, we discuss the results of our study with respect to our research questions. As previously stated, our interviews revealed that three sources of information are useful during trace link vetting: 1) information from the connected artifacts, 2) information from the traceability metamodel, and 3) information from the tracing algorithm. We designed an experiment to validate how context information affects the analyst's performance when vetting trace links. The context information we studied came from the first source of information i.e., the connected artifacts. Our experiment showed that there is no statistically significant difference between the group that was provided with context information and the group that was not. The experiment also showed that the analysts spent more time investigating the content of the artifacts

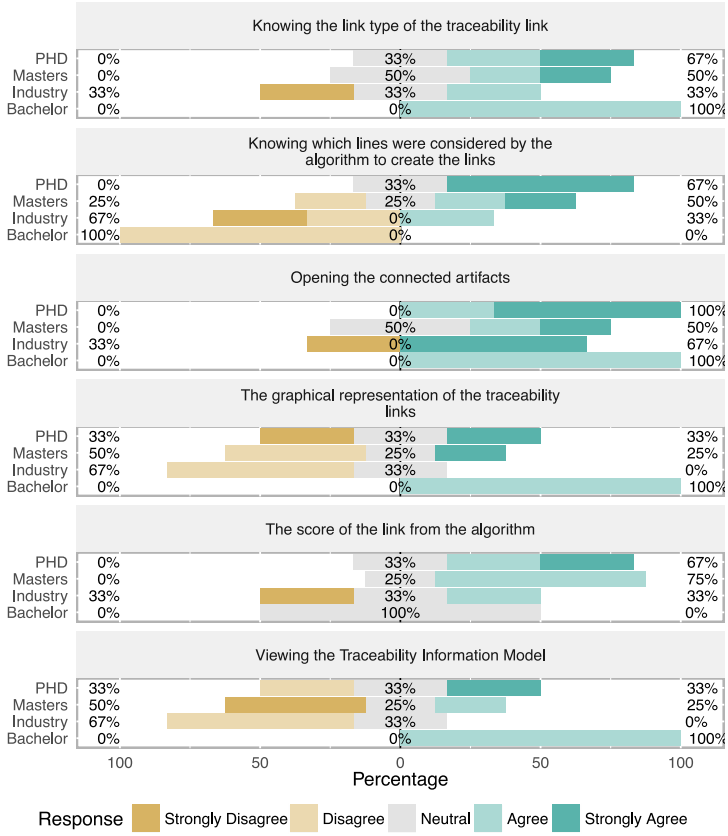


Figure 5.4: Comparison of Industry participants vs. student participants

than investigating the context information provided. This could be due to unfamiliarity with the system and its context, analysts tried to understand the artifacts by looking at their content rather than their context information. This behaviour of the analyst is also supported by what was reported by the interviewees. A majority reported that familiarity and experience with the system is important when vetting trace links.

While there are some differences between industry and student participants (cf. Table 5.6), the difference is not statistically significant. We believe that it is experience with the particular system that plays a larger role in the performance of the analyst instead of general software development experience. The study by Dehtyar et al. [42] also shows neither experience with software development nor tracing experience had an influence on the analysts' performance. *We therefore suggest that analysts who have system experience be assigned to the task of vetting traceability links.* Developers should, e.g., be assigned to vet links between requirements and code and safety analysts should be assigned to vet links between requirements and faults. However, further research is needed to support this.

Regarding the second source of information, i.e., the traceability information model, participants from both the control and the experiment group thought

that knowing the link type of the trace link was useful, but having the entire TIM visible to them at all times was not very useful. This could be because the experiment only included four types of elements (requirements, code, assumptions, and faults) and therefore the TIM was small. The analysts only had to see the TIM once to understand which links were possible. The information from the third source, i.e., the tracing algorithm, such as the similarity score was also perceived as useful by the experiment participants which is in line with the study by Hayes et al. [45].

5.9 Threats to Validity

In this section we discuss the threats to validity that are relevant to our study. Since we used multiple research methods we discuss the threats in a combined manner.

Construct Validity We selected interviewees who had experience with tracing and explained the study prior to the interview. For the experiment, a preliminary session introduced traceability and the aim of the study. However, some participants did not follow the instructions of the experiment. To limit this threat to construct validity, we viewed the screen videos of all participants to exclude these participants.

Table 5.6: Performance of industry subjects vs. student subjects

Industry Subjects					
Subject	Accepted	Rejected	Investigated	Precision	Recall
A	54	81	135	19	56
B	64	120	184	16	50
C	111	49	160	10	61
D	46	92	138	20	50
Average	**68.75**	**85.5**	**154.25**	**16.25**	**54.25**
Student Subjects					
A	49	44	98	20	67
B	12	2	19	25	100
C	17	67	84	47	53
D	47	84	134	21	56
E	103	56	159	16	89
F	42	0	42	24	100
G	73	36	109	19	82
H	210	113	323	11	92
I	17	19	36	29	56
J	47	7	54	32	100
Average	**61.7**	**42.8**	**105.8**	**24.4**	**79.5**

Internal Validity The aim of the experiment was to evaluate if the context information makes the analyst more effective. There are four confounding factors that could affect the results of the experiment: tracing experience of the analyst, implementation of link vetting features in Eclipse Capra, the tracing strategy used by the analyst, and the system used for the experiment (MedFleet). While the study by Dekhtyar et al. [42] shows that tracing experience has no effect on the performance of the analyst, it also used students with limited tracing experience. In our study, only eight of the experiment participants had experience with tracing. Conducting the experiment with traceability experts could lead to different results. Regarding Eclipse Capra, we made sure that we only implemented features that were requested by the interviewees and also evaluated existing tools to see how these features were previously implemented. Regarding the tracing strategy, the experiment participants were asked to vet the list of trace links from top to bottom. This ensured that the analysts did not skip any links without investigating them. However, the tracing strategies still varied as some analysts skipped the links they did not understand immediately, while others investigated the link for a longer time before making the decision of skipping the links. This resulted in a variation on the number of links investigated.Regarding the system used for the experiment, we selected a system containing artifacts that are as close to reality as possible. Even though MedFleet was developed by students, it was in a course where students learned software development skills and had to follow proper software development procedures just like in industry. However, this does not guarantee that the system selected had no effect on the study. To properly rule out this internal validity threat, further experiments are needed with different systems.

External Validity We took several steps to ensure that our study included diverse participants. We interviewed practitioners with different roles, from different companies, and different countries. However, since not all interviewed participants had experience with trace links vetting, we cannot generalize that we elicited all possible context information. Additionally, since the number of interviewees we had was low, we cannot generalize the results. In the experiment, we used participants from different companies and two different universities, with different levels of education and development experience.

Reliability We documented our process in the research method section (Section 5.3) and have also published our interview guide to make sure that our study can be repeated. Our prototype tool is also accessible as a virtual machine, meaning that the experiment can be repeated by other researchers.

5.10 Conclusion

In this study, we investigated what information is useful to human analysts when vetting automatically generated traceability links. We specifically investigated if context information can improve the analyst's vetting performance when vetting traceability links. Our interview results reveal three sources of information that can be useful to the analyst. However, our experiment presents evidence to support the conclusion, though not statistically significantly, that context

information does not make the analyst more effective during the vetting process. The experiment also shows that analysts made their decisions on trace links by reading the content of the artifacts rather than using context information. Our study shows that the experience of the analyst with the particular system matters more than their general software development experience or tracing experience. We conclude that since the vetting process is a human-centric process, further studies are needed to investigate how this process can be improved by traceability tools to make the analyst more effective.

Acknowledgements

Funding for this work has been partially provided by the US National Science Foundation under grants CNS-1649008, CCF-1647342, and CCF-1511117.

Chapter 6

Paper E

Impact of Gamification on Trace Link Vetting: A Controlled Experiment

S. Maro, E. Sundklev, C-O. Persson, G. Liebel, J.-P. Steghöfer

In International Working Conference on Requirements Engineering: Foundation for Software Quality, pp. 90-105. Springer, Cham, 2019.

Abstract

[Context] Automatically generated trace links must be vetted by human analysts before use. The task of vetting trace links is considered boring due to its repetitive nature and tools that are not engaging to the analyst. Therefore, a lack of developer engagement can hamper the successful implementation of a traceability strategy in an organisation. **[Objective]** In this study, we examine whether two gamification features, levels and badges, have a positive effect on human analysts' engagement and ultimately on the quality of vetted trace links. **[Method]** We have conducted a controlled experiment with 24 participants that vetted trace link candidates and recorded their speed, correctness, enjoyment, and perceived usability of the tool. **[Results]** The results indicate that there was no significant difference between the speed, correctness, and perceived usability of the control and the experiment group. However, gamification features significantly increased the users' perceived enjoyment. Levels and badges were perceived positively by the majority of the participants while some pitfalls and improvements were pointed out. **[Conclusion]** Our study indicates the need for further research as the results raise several questions, in particular w.r.t. what analyst behaviour gamification incentivises, and the impact of gamification on long-term enjoyment.

6.1 Introduction

Traceability is important in the software industry as it aids both developers and managers in maintaining the relationships between software artefacts such as requirements, design, code, and documentation. Traceability is also required by certain safety standards, such as ISO 26262, or to obtain certification for organisational maturity, e.g., when using Capability Maturity Model Integration (CMMI). Creating and maintaining trace links is cumbersome when the systems involved are large and contain a large number of artefacts. To reduce the effort of creating and maintaining trace links, information retrieval approaches [25] such as machine learning [21] have been proposed to automatically generate trace links. However, since automated approaches produce a relatively high number of candidates that are not valid links, a human analyst needs to vet candidates before they become actual trace links. This task of vetting trace links is perceived as boring by many analysts [220].

Previous studies, e.g., Kong et al. [207] and Dekhtyar et al. [42], investigate how analysts vet trace links and how this process can be improved. However, none of these studies have investigated how to make the vetting process more engaging and enjoyable to the human analyst.

When attempting to engage users, gamification has shown to have a positive motivational effect [221–223]. Additionally, gamification has been shown to reduce the rate of failure and assists in the learning process in some areas [224]. Specifically for traceability, Parizi [225] showed that gamification concepts improve the task of creating trace links between code and tests during software development. She also points out that gamification elements could be useful for other human-centered tracing activities, such as vetting automatically generated trace links.

To address the lack of studies about the impact and potential benefit of applying gamification to traceability task, we investigate the effects of gamification on vetting automatically generated trace links. Specifically, we investigate the effect of two gamification features – levels and badges. Concretely, we aim to answer the following research question:

RQ: What is the impact of gamification on the task of vetting automatically generated trace links?

To answer these questions, we conducted a controlled experiment in which we asked 24 participants to vet automatically generated trace links. Twelve participants used the traceability management tool Eclipse Capra [69] without modifications, while the remaining twelve participants used the same tool extended with gamification elements. We investigated the impact of gamification on the total number of links vetted, the accuracy of vetted links, on the motivation of the vetting task, and on the perceived usability of the tool.

Our results show no significant difference between the two groups with regards to the final precision and recall of the vetted links, total number of vetted links and usability of the tool. However, the results show that gamification elements have the potential to increase enjoyment and motivate the users for such a task.

The remainder of this paper is structured as follows: In Sect. 6.2, we discuss the background as well as similar studies on trace link vetting and gamification

in software engineering. We then describe our methodology in Sect. 6.3 and our results in Sect. 6.4. Sect. 6.5 provides answers to our research questions before we conclude the paper in Sect. 6.6.

6.2 Background and Related Work

This section discusses the background of our work and related studies. We discuss trace link vetting and the use of gamification elements in software engineering.

6.2.1 Vetting Automatically Generated Links

Automatically generated trace links are flawed and need a human analyst to vet them for correctness and completeness. Hayes et al. [226, 227] discovered that the human analyst can make the generated set of trace links worse. Since then, several studies have been conducted to better understand and to improve the process of vetting trace links. In a controlled experiment, Cuddeback et al. [36] showed that human analysts decreased the quality of a high-quality set of initial trace links, while they increased the quality if it was initially low. In other experiments, Cuddeback et al. [43] and Kong et al. [228] confirm these findings.

To understand different strategies used by human analysts in vetting links, Kong et al. [207] studied logs from a link-vetting experiment and identified strategies such as "accept-focused", where the analyst only accepted links, and "first good link", where the analyst focused on finding the first good link. Additionally, Hayes et al. [78] conducted a simulation study on the different vetting strategies to understand which strategy was the most effective. The authors show that analysts have the best performance when they examine a list of top candidate links that has been pruned based on some heuristics to remove low ranked links, and if the tool takes the analyst's feedback into consideration to modify the list of candidate links dynamically.

Dekhtyar et al. [42] conducted an experiment to understand how factors such as the development experience, and tracing experience of the analyst affect their performance when vetting trace links. The authors showed that development experience, tracing experience, effort used to search for missing links, and how prepared the analyst felt had no significant influence on the performance. However, they also show that the self-reported effort used on validating trace links had a significant effect on the performance: analysts that spent a lot of time validating links ended up reducing recall by rejecting correct links.

While these existing studies investigate the performance of the human analyst, there exist to our knowledge no studies dedicated to improve the trace link vetting process by making it more engaging.

6.2.2 Gamification in Software Engineering

Several studies have investigated how to incorporate gamification elements in software engineering tasks for the purpose of increasing engagement and motivation of people performing different tasks. Pedreira et al. [221] published

a systematic mapping study that shows the distribution of gamification studies in software engineering. The authors show that most studies focus on the software implementation task (coding), followed by project management and process support, while only few studies targeted requirements engineering and software testing. Since the publication of this mapping study, more studies have been published in the area of requirements engineering, e.g., in requirements elicitation [229, 230] and requirements prioritization [231, 232].

Additionally, there exist a number of studies on a meta-level, focusing on how gamification can best be introduced in software engineering. These studies, for example Kappen and Nacke [233], Morschheuser et al. [234] and Garcia et al. [235] define guidelines and frameworks to guide software engineers on how to effectively gamify software engineering activities. Our research method is in line with these frameworks that suggest analyzing the activity to be gamified, implement the gamification features, and evaluate the impact of the features.

We are aware of only one study that targets traceability and is therefore related to our study. Parizi [225] investigated the impact of gamification when tracing between code and tests. The authors conducted an experiment showing that use of gamification elements, namely case points, feedback, reputation, avatars, progress and quests, improved both precision and recall of the recovered set of manually created trace links. The gamification elements encouraged the developers to create more links. In this paper, we study a different phenomenon where gamification elements are applied to encourage the human analyst to vet trace links based on candidates created automatically.

6.3 Research Method

In order to answer our research question, we conducted a controlled experiment [236], comparing vetting of automatically generated trace links with and without gamification.

6.3.1 Experiment Design

We used a simple one-factor experiment design with two treatments [236], namely the use of the traceability software without gamification and the use of the same software with gamification. We refer to the subjects using the software without gamification as our *control group*, while the subjects using the gamified software are in the *experiment group*. The dependent variables are the overall number of vetted trace links (*vetted*), the fraction of correctly vetted trace links (*vettedCor*), the self-reported motivation throughout the experiment (*motivation*), and the impact of gamification on the perceived usability (*usability*). The variables *vetted* and *vettedCor* can furthermore be divided into accepted and rejected trace links, i.e., *accepted*, *rejected*, *acceptedCor*, and *rejectedCor*. We derive the following null hypotheses and corresponding alternative hypotheses:

- $H0_{vetted}$: There is no significant difference in *vetted* between the control group and the experiment group.
- $H1_{vetted}$: There is a significant difference in *vetted* between the control group and the experiment group.

- $H0_{vettedCor}$: There is no significant difference in *vettedCor* between the control group and the experiment group.
- $H1_{vettedCor}$: There is a significant difference in *vettedCor* between the control group and the experiment group.
- $H0_{motivation}$: There is no significant difference in *motivation* between the control group and the experiment group.
- $H1_{motivation}$: There is a significant difference in *motivation* between the control group and the experiment group.
- $H0_{usability}$: There is no significant difference in the perceived *usability* between the control group and the experiment group.
- $H1_{usability}$: There is a significant difference in the perceived *usability* between the control group and the experiment group.

To decide what kind of gamification elements to implement, we sent out a survey to 14 subjects that had participated in a previous study using trace link vetting with Eclipse Capra, reported in [220]. For each candidate gamification element, we asked the subjects a number of questions pertaining to the potential enjoyment and distraction caused by the element. We received 11 responses to our survey. The results and implementation of the gamification elements are described in Sect. **??**.

We used MedFleet as an instrument, a drone fleet coordination system which contains requirements and fault descriptions as well as source code. A manually created set of trace links served as a ground truth to which we could compare the vetting results. We generated trace link candidates using Vector Space Model with Term Frequency – Inverse Document Frequency (TFIDF), a technique that is commonly used to generate links between textual artifacts [237].

During the experiment, we asked participants to work through a list of candidate trace links between any of the three artefact types. For each link, participants had to decide whether the two linked artefacts indeed have a relation to each other. If yes, the candidate link should be *accepted*, if not, then it should be *rejected*. To support this process, Eclipse Capra offers the ability to open and view the artifacts the candidate link refers to, including the source code files. Prior to the experiment, we handed out a written document describing all relevant features of Eclipse Capra. For the experiment group, this description also contained an explanation of the gamification elements.

6.3.1.1 Data collection and analysis

The participant sample for this experiment consisted of 24 students with an academic software engineering background. We assigned the participants randomly into balanced experiment and control groups. While all subjects were students, some had experience as software developers. Only eight of the students, four in the control group and four in the experiment group, had experience with traceability. Additionally, all students had some experience with using Eclipse. The use of student subjects in experimentation has been debated heavily and controversially [73]. As this is an initial study, we believe

that the use of student subjects is favourable over practitioners due to the higher homogeneity in their knowledge and thus higher internal validity [73]

After the introduction to Eclipse Capra, participants had 45 minutes to complete their task. Using the features of Eclipse Capra freely, we encouraged participants to accept and reject as many candidate links as they could, while taking the time they needed for decision making. Participants were allowed to ask questions regarding the operation of Eclipse Capra and the gamified system during the entire experiment.

We instrumented Eclipse Capra to monitor participant activity, namely accept and reject events, as well as opening events of artifact files. Additionally, we collected data through a pre-experiment questionnaire (collecting demographic data) and a post-experiment questionnaire (collecting perceptions about gamification and system usability scale (SUS) [71] scores). All survey instruments as well as the instructions are available online [238].

6.3.2 Validity threats

There are several potential threats to validity in this study.

To avoid survey questions being misinterpreted by participants, we ran each questionnaire through several internal review rounds, identifying and improving potentially ambiguous questions.

Domain knowledge about the MedFleet system can affect the correctness of trace links per participant. While none of the participants had any prior knowledge of MedFleet, and therefore an advantage in domain knowledge, this lack of domain knowledge could also threaten the external validity of the study. We accepted this potential threat in favour of having a higher internal validity due to the homogeneity of the student population.

Individual differences between participants could pose another threat to validity, as we did not use a crossover design. Since we only had a single software system with all the required artefacts (requirements, source code, faults, and traces between them), we had to accept this potential threat.

Given that gamification features were added to Eclipse Capra, there is a potential threat that these modifications affected the usability of the system and, hence, confounded the results. Additionally, the specific implementation of the features could have an effect on the results. To assess this, we collected usability information in the post-experiment survey as discussed in Sect. **??**.

As for all empirical studies, there is a trade-off between internal and external validity [73]. We opted for a higher internal validity, e.g., by choosing student subjects. This naturally limits the generalisability of our results. For this initial work, we believe that this restriction is acceptable, but at the same time encourage replications with a more diverse sample of participants.

6.4 Results

We first explain the two gamification features we chose to test in our experiment — levels and badges. We then describe the results of our experiments, including answers from the questionnaires as well as the analysis of vetting accuracy.

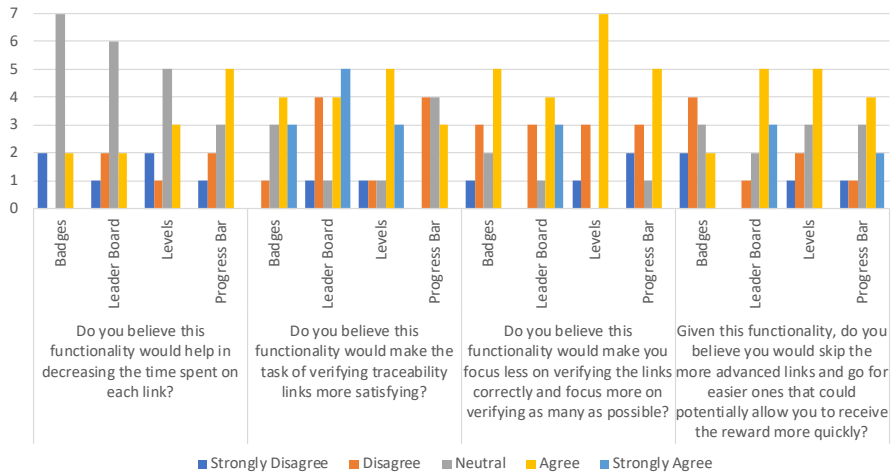
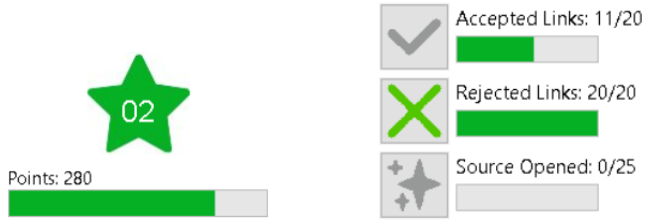


Figure 6.1: Pilot survey results showing aggregated responses for the gamification features.

Gamification features We used the responses from the pilot survey (Sect. 6.3) to decide on the gamification features to be implemented and tested in the experiment. Out of the four suggested gamification features, we selected *levels* and *badges* as the most viable options. As can be seen in Fig. 6.1, the results of the pilot survey were very mixed across the different features. While the *progress bar* received the most positive results in terms of potential decrease of the time spent on each link, some commentators also pointed out that a progress bar that does not fill up quickly, e.g., because of the large number of links to vet, can be discouraging. In addition, there were participants that felt that they would optimise towards vetting as many links as possible and potentially skip difficult ones. *Leader boards*, on the other hand, received high scores in the ranking for how satisfying they would make the task, while also showing a high spread. However, there are indications from the free text comments that leader boards can be perceived as too competitive, even though findings from the literature show that competitiveness might have positive effects on performance [239]. In addition, there were respondents who strongly agreed that they would again favour number of links over vetting more complicated ones. For both *badges* and *levels*, no indications could be found that they would decrease the time spent on each link, but both have good values for increase in satisfaction. While there are indications that levels would shift focus to vetting as many links as possible, there is no strong indication that users would skip more complicated links.

Both implemented features are shown in Fig. 6.2. The current *level* of the user is shown within the green star next to the total points accumulated and how much progress is left until the next level is reached. This is different from the progress bar suggested as a feature since it does not measure progress of the overall vetting task, but still provides feedback on the progress towards the next level. The levelling system awards 10 points for accepting or rejecting a candidate link. The next level is reached after 100 points have been collected.



(a) (b)

Figure 6.2: The level and badges as shown in the modified version of Eclipse Capra.

Table 6.1: Experience with systems similar to the experiment system and confidence of understanding the experiment system.

		Control Group	**Experiment Group**
Experience with systems similar to MedFleet	Yes	0	2
	No	12	10
Confidence of understanding the MedFleet system	Strongly disagree	0	0
	Disagree	2	2
	Neutral	3	6
	Agree	6	3
	Strongly Agree	1	1

We integrated three different *badges*: one is awarded after accepting 20 links, one after rejecting 20 links, and one after opening 25 source code files. The figure shows an icon for each badge, how many links have been accepted or rejected, how many source code files have been opened, and how much progress has been made for each badge. When the requirements of a badge are fulfilled, the logo turns green as can be seen on the "rejected links" badge.

Experience with and understanding of experiment system The post-experiment part of the survey contained some questions common for both groups. Two questions aimed at understanding previous experience with systems similar to the one used in the experiment and the confidence that participants felt in working with the system. The answers to these questions allowed us to gauge if any of the groups had an advantage over the other due to previous experience or a vastly higher confidence in working with the system. The responses in Table 6.1 show, however, that no significant differences exist. Only two of the participants, both in the experiment group, had prior experience in similar systems and confidence levels for understanding the system are very similar.

Participant enjoyment and motivation To understand the impact of the gamification features on the enjoyment of the participants and on their motivation to complete the vetting task, all participants were asked explicitly about these aspects in the post-experiment questionnaire. As can be seen in

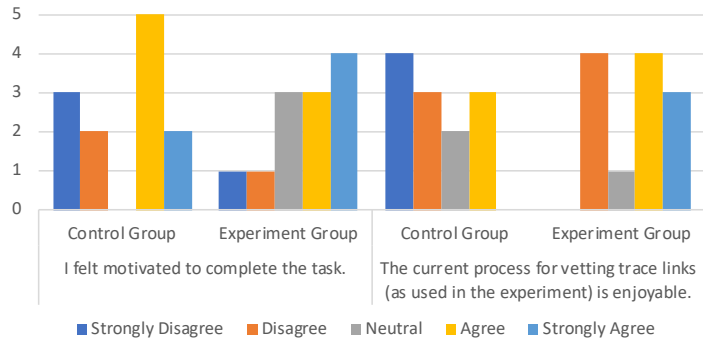


Figure 6.3: Results of the post-experiment survey showing aggregated responses for enjoyment of the task and motivation to complete the task.

Fig. 6.3, the experiment group shows a tendency towards finding the vetting process more enjoyable and feeling more motivated to complete the task. The Mann-Whitney u-test on the responses shows that the results are statistically significant ($p \approx 0.040$), thus corroborating $H0_{motivation}$. This result shows that adding gamification features is beneficial to the enjoyment of vetting the links and to the motivation of completing the task.

System usability scale In order to compare the usability of the gamified and the regular versions of Eclipse Capra, we used the System Usability Scale (SUS) [71]. It contains a total of ten statements which are answered on a five-point Likert scale. From the responses, we also computed the overall SUS score. For this purpose, we deducted 1 from the score for each positively-worded statement (statements 1, 3, 5, 7, and 9) and deducted the score of each negatively worded statement (statements 2, 4, 6, 8, 10) from 5. The resulting scores were multiplied by 2.5 and added up to achieve a range from 0 to 100. The results are shown in Table 6.2.

According to Bangor et al. [240], SUS scores between 0 and 25 are considered *worst imaginable*, scores between 26 and 38 are considered to be *poor*, scores between 38 and 52 are considered to be *OK*, scores between 52 and 73 are considered to be *good*, scores between 73 and 85 are considered to be *excellent* and finally scores between 85 and 100 are considered the *best imaginable*. Therefore, the scores provided both by the control group and the experiment group are in the *good* range with a slight advantage for the gamified version. The results from a Mann-Whitney u-test on the average score from both groups showed to be insignificant ($p \approx 0.904$). The SUS-scores show that the usability of Eclipse Capra is *good* with and without extension with level and badge features. We can thus reject $H1_{usability}$ and conclude that there is no significant difference in usability.

Attitude towards levels and badges To gauge the attitudes of the participants towards the gamification features, we asked both groups different questions about their perceptions. While the experiment group was asked about how they rated their experience, the control group was given a brief

Table 6.2: SUS for the control group and the experiment group. For each question, the average values in the groups are depicted, with standard deviation in brackets.

		Control Group	**Experiment Group**
Q1	I think that I would like to use this system frequently	2.25 (1.55)	3.00 (1.05)
Q2	I found the system unnecessarily complex	2.33 (1.37)	2.08 (1.08)
Q3	I thought the system was easy to use	3.83 (0.94)	4.17 (0.58)
Q4	I think that I would need the support of a technical person to be able to use this system	1.92 (1.08)	2.25 (1.22)
Q5	I found the various functions in the system were well integrated	3.66 (0.89)	3.75 (0.86)
Q6	I thought there was too much inconsistency in the system	1.75 (0.75)	1.92 (1.08)
Q7	*I would imagine that most people would learn to use this system very quickly*	4.08 (1.16)	4.08 (1.00)
Q8	I found the system very cumbersome to use	2.25 (1.60)	2.17 (0.94)
Q9	I felt very confident using the system	3.00 (0.95)	3.50 (1.68)
Q10	I needed to learn a lot of things before I could get going with this system	2.00 (1.21)	2.25 (1.14)
SUS score		66.56	69.58

demonstration of the gamification features after they completed their task and were then asked about how these features might have influenced their experience. We were also interested in gauging whether the participants thought that there was an impact on the vetting process or on their individual performance. All questions were answered on a five-point Likert scale. The attitudes of the experiment group are shown in Table 6.3 and the attitudes of the control group in Table 6.4.

The averages in the tables show no major difference between the levels and badges features. The majority of the participants understood the features, but it was considered easier to understand the badges which is overall the biggest difference between the two features. The control group on average thought that the levels feature would make them focus less on correctness and more on verifying as many links as possible, at least compared to the badges feature.

Vetting task results This section presents the results from the vetting task that all the participants undertook. The results from both groups can be seen side-by-side in Table 6.5. On average, participants in the experiment group vetted 130 links, while participants in the control group vetted 160. The average rate of correctly accepted links for the experiment group was 16.54%

Table 6.3: Attitude of the experiment group towards the levels and badges feature

	Levels	**Badges**
I had no issue understanding the levels/badges	3.4 (1.31)	4.2 (1.11)
The levels/badges made the task of vetting trace links satisfying	3.6 (0.90)	3.8 (0.94)
The levels/badges helped decrease the time spent on each link	3.2 (0.94)	2.9 (0.90)
The levels/badges made me focus more on verifying as many links as possible instead of verifying each link correctly	3.1 (1.17)	3.1 (1.38)
I felt that the levels/badges contributed towards being motivated to complete the task	3.4 (1.08)	3.5 (1.24)
Overall, the levels/badges feature was a good addition to the traceability tool	3.9 (0.67)	3.8 (0.94)

Table 6.4: Attitude of the control group towards the levels and badges feature

	Levels	**Badges**
Levels/badges make the task of verifying trace links more satisfying	3.4 (1.24)	3.6 (1.62)
Levels/badges help in decreasing the time spent on each link	2.75 (1.49)	2.7 (1.16)
Levels/badges make me focus less on verifying the links correctly and more on verifying as many as possible	3.4 (1.38)	2.8 (1.22)
Levels/badges would make me skip the more advanced links and go for easier ones in order to "level up"/receive more badges faster	3.4 (1.24)	3.2 (1.53)

and 17.19% for the control group. The average rate of correctly rejected links for the experiment group was 92.14% and 92.15% for the control group. This is reflected in the measures for precision and recall that show a low precision of 0.17 for both groups, but a recall of 0.62 and 0.67, respectively. This indicates that detecting correct links was harder compared to rejecting wrong links that were obvious to spot. Standard deviations for the total number of links as well as for precision are relatively high, indicating that there was a relatively large spread in the rate of true positives amongst all selected positives. Indeed, the control group, e.g., ranged between a correctness of 7.7% and 52%.

We tested our null hypotheses $H0_{vetted}$ and $H0_{vettedCor}$ with a Mann-Whitney u-test, yielding p-values of $p \approx 0.542$ and $p \approx 0.912$, respectively. Therefore, we can not reject the null hypotheses and can not detect a statistically significant difference in either the number of vetted links or the correctness of vetting decisions between the experiment group and the control group.

Table 6.5: Results of the vetting task for the control group and the experiment group.

	True positive	False positive	Total positive	Rate	True negatives	False negatives	Total negatives	Rate	Precision	Recall
Control Group										
Avg.	9	66.92	75.92	17.19%	77.58	5.75	83.33	92.14%	0.17	0.62
Stdev.	2.26	43.79	43.33		57.83	3.28	60.25		0.14	0.19
Experiment Group										
Avg.	9.67	60	69.67	16.54%	56	4.67	60.67	92.15%	0.17	0.67
Stdev.	2.71	30.84	31.92		21.34	2.46	22.23		0.08	0.18

6.5 Discussion

In this section we discuss the implications of the results and how they relate to our research question. To recap, our research question is: *What is the impact of gamification on the task of vetting automatically generated trace links?*

Our results show that, on one hand, there is no significant difference in the total amount of links vetted and also in the accuracy of the vetted links between the experiment and the control group. From the survey carried out to elicit which gamification features are suitable, there were indications that gamification might lead to participants vetting more links but reduce accuracy. However, our results show that this is not the case. We would like to emphasise that this is not a negative result. In contrary, the results mean that the newly added gamification elements did not lead to participants rushing the vetting task in favour of reaching higher levels or winning badges. Indeed, it was difficult for participants to know which accuracy they attained since there was no feedback as to whether the links accepted or rejected were correct. For the future, we consider alternative means of rewarding correctness, such as rewarding taking extra time to analyse an artefact in detail. Another alternative proposed by an experiment participant is to allow the participant to see the vetting decisions of other users before making their own. Decisions of other users would allow the current user to get an idea of what others thought was correct, but is by no means guaranteed to be correct. Since reward options are limited if correctness can not be checked automatically, the usefulness of gamification for trace link vetting can be restricted. Agreement with other users might be used as a stand-in, but might lead to skewed results when users are incentivised to agree with others.

The design of the badges and the associated reward system could have an influence on the precision achieved by the participants. In our case, participants felt the need to accept approximately as many links as they rejected since both actions were associated with the reward of badges. The low number of correct links among the link candidates might have contributed to the low correctness rate of accepted links. To better understand this factor, more research is needed with different types of badge designs and reward systems.

The results show that the experiment group found the vetting task to be more enjoyable and motivating than the control group. Many studies on gamification consider how and if a gamified implementation has had an impact on intrinsic and extrinsic motivation (see, e.g., [222, 241, 242]). Intrinsic motivation can be described as being motivated to perform a task because one enjoys doing it, while extrinsic motivation can be described as completing a task because of the incentive one gets after completing it [243]. When attempting to motivate people, it is considered best practice to aim at increasing intrinsic motivation, since being motivated to do something because of enjoyment is more sought after than being motivated by extrinsic rewards [243]. In our case, since participants of the experiment group reported that they enjoyed the task and felt more motivated to perform it, but did not increase the number of links or their accuracy, there is an indication that the participants had more intrinsic than extrinsic motivation. However, we cannot conclude that this is the case and further research is needed to investigate this motivational aspect by, e.g., letting participants choose if they want to perform the vetting task or not and

observing their performance over time.

Our findings are also inconclusive w.r.t. the long-term effects of gamification. Since participants were only exposed to the gamification features for one session, we can conclude that within this session, enjoyment and motivation was increased, but can not state that these positive effects would be present over a longer period of time and a number of consecutive sessions. There is a chance that participants get used to the features and their level of enjoyment and their motivation decreases over time. This can be counteracted with a reward system and gamification features that keep users engaged continuously, e.g., with specific rewards for higher levels. Our experiment is only an initial step and more research is required to understand the long-term motivational aspects in more detail.

6.6 Conclusion

In this paper, we investigated the impact of gamification on trace link vetting. We identified suitable gamification features based on existing studies and a survey, and implemented levels and badges in the traceability management tool Eclipse Capra. To test the impact of these features, we conducted a controlled experiment with 24 student participants, comparing the use of Eclipse Capra with and without gamification features. Specifically, we investigated the impact of having levels and badges on the correctness and the number of vetted links, as well as the perceived motivation of the participants and the usability of the tool.

The results show that our implementation of levels and badge features had no significant effect on the correctness and amount of vetted links. However, the participants found the gamified system to be more enjoyable. Furthermore, participants did not identify differences in the usability of the gamified and the non-gamified system. The results of our initial survey also showed a difference in participants' preferences toward competitive gamification elements (e.g., leader boards) compared to non-competitive elements (e.g., levels).

For future work, we see a number of possibilities. First, it is essential to study the long-term effects of gamification on enjoyment or motivation, and how to keep subjects engaged over longer periods of time. Our experiment only serves as an initial indication that gamification indeed increases motivation in the short term, while it remains to be studied whether this effect wears off over time. Second, we see the need to replicate our study in an industrial setting with professional developers. Since the experiment setup requires a ground truth of which traces are correct and which ones are not, this is a challenging task. Finally, there is a need to study how a more advanced implementation of levels and badges affect the vetting task since in this study we only test our specific implementation.

Chapter 7

Paper F

TracIMo: A Traceability Introduction Methodology and its Evaluation in an Agile Development Team

S. Maro, J.-P. Steghöfer, P. Bozzelli, H. Muccini

Under revision at the Requirements Engineering Journal.

Abstract

Traceability is an important aspect in software development that yields a number of benefits such as facilitating impact analysis and tracking software changes. However, for companies to reap these benefits, a proper traceability strategy needs to be defined and implemented. Existing literature lacks concrete guidelines for practitioners to systematically define such a strategy. In this study, we address this gap by defining TracIMo, a methodology for systematically designing and introducing software traceability in practice. We used design science research to define and evaluate TracIMo with a case study in an agile development team of a company in the finance domain. Our results show that TracIMo is feasible as it allows incremental definition and evaluation of a traceability strategy that is aligned with the company's traceability goals and the existing development process. The evaluation showed that the resulting traceability strategy yields a number of short-term benefits, including improvements in effort estimation and understandability of tasks. Additionally, this paper reports practical challenges encountered when designing a traceability strategy.

7.1 Introduction

Traceability is defined as "the ability to interrelate any uniquely identifiable software engineering artifacts to any other, maintain required links over time, and use the resulting network to answer questions of both the software product and its development process" [11]. Software engineering artifacts include artifacts such as requirements, design models, implementation, and tests as well as process-related artifacts such as tasks and tickets. Traceability is an important aspect in software development, providing benefits such as supporting change impact analysis [141,244], program comprehension [167] and compliance to standards [245]. Even with all the promised benefits, many companies developing software lack systematic traceability strategies [83]. Trace links are created and maintained in an ad hoc manner and therefore benefits are not visible due to the mismatch in the established strategy and the traceability needs of the company [246]. A traceability strategy is a plan of action for how traceability should be established and maintained in an organisation. The strategy defines how traceability activities such as creation, maintenance and use of traceability should be conducted. This includes defining the purpose of traceability and how it should be managed both in terms of tools and processes [10].

One of the reasons for ad hoc traceability is the lack of concrete guidelines for practitioners on how to establish traceability [83]. This can lead to effort invested in creating and maintaining trace links which are ultimately inconsistent, incomplete and never used [247]. While there is literature reporting on case studies in which traceability is established (see, e.g., [17, 248]), these studies do not give concrete guidelines that are still generic enough so they can be easily transferred to other cases. Other studies e.g., Dömges et al. [18], give abstract descriptions on how to establish and maintain project-specific traceability that are not directly actionable in practice. Moreover, Cleland-Huang et al. explicitly point to the lack of guidance for practitioners when establishing traceability in their paper discussing future research directions for traceability [34]. An attempt toward addressing a related, but distinct problem is the study by Rempel et al. [246], which provides a framework for assessing an existing traceability strategy in companies, in particular the alignment of the strategy with the traceability needs at the company. This work however, does not give concrete guidelines to follow when traceability is not yet established.

The aim of our contribution is therefore to extend the state-of-the-art by defining a methodology for systematically designing and deploying company specific traceability strategies. We used design science to design and evaluate our methodology, called TracIMo, short for Traceability Introduction Methodology, in collaboration with an agile development team in the finance domain.

With this study we answer the following research question:

RQ: How can traceability be established to achieve the goals of the organisation and yield a measurable impact?

The contribution of this paper is threefold: first, we present a structured methodology for designing a traceability strategy and introducing traceability in a software development organizations. It was inspired by and extended from the traceability strategy assessment procedure in [246]; second, we discuss

benefits of the introduced traceability strategy; and third, we discuss challenges and important decisions that need to be made when designing a traceability strategy in order to maximize its benefits. Our aim is to give both researchers and practitioners practical insights into how to establish traceability.

The rest of the paper is structured as follows: In Section 2 we discuss previous studies related to our work. Section 3 describes our research methodology. In Section 4 we describe the methodology to introduce traceability (TracIMo) while Section 5 describes how it was evaluated at the company. Section 6 gives a discussion with respect to the research question and Section 7 compares TracIMo with similar existing frameworks and methodologies. Section 8 describes the threats to validity of our study and Section 9 concludes the paper and outlines future work.

7.2 Related work

A vast amount of traceability research covering various topics is available. We performed a lightweight systematic mapping study by searching ten top software engineering publication venues for traceability research (see Table 7.1) using SCOPUS. We searched for papers that specifically mentioned the term “traceability" in the title and published in the respective venues. For instance, the search term used to search the requirements engineering conference was as follows:

TITLE(“Traceability”) AND CONFNAME(“Requirements Engineering”).

The search was performed on July 2nd and we did not restrict the publication time of the papers. We screened the identified papers by reading the title and abstract specifically looking for papers suggesting frameworks or methodologies for introducing traceability. We also screened for papers reporting industrial case studies of introducing traceability. This mini-review also allowed us to identify the main topics of research w.r.t. traceability published in these venues.

Overall the main topics of traceability are research on specific tools and technologies for managing traceability (e.g., [249, 250]), automation of trace link creation using information retrieval and machine learning approaches (e.g. [21, 24, 251]), traceability between specific artifact types, e.g., business models to architecture [252], requirements to code [253], as well as research that leverages model-based development techniques for traceability e.g. [247, 254]. A large amount of this research is evaluated using example systems from universities (e.g., [250]) or using example systems taken from industry which are not publicly available (e.g., [24]). There are only few studies where approaches are evaluated on running industrial projects (e.g., [255]).

Specifically, there is little research describing how to define traceability strategies. In practice practitioners struggle with defining traceability strategies suitable to their specific company needs [34]. In this area most of the research is focused on designing traceability strategies to support development of safety-critical products since traceability is mandated by safety standards. For instance, Nair et al. [256] provide an overview of traceability for safety evidence certification. They discuss what the goals for traceability of safety evidence are (e.g., safety assurance and change impact analysis) and propose a traceability

Table 7.1: Publication venues and the number of papers on software traceability.

Venue	No. of papers
Requirements Engineering Conference (RE)	84
Requirements Engineering Foundations for Software Quality Conference (REFSQ)	13
International Conference of Software Engineering Conference (ICSE)	37
Foundations of Software Engineering Conference (FSE)	9
Transactions of Software Engineering Journal (TSE)	9
Journal of Software and Systems (JSS)	17
Information Software Technology Journal (IST)	14
Requirements Engineering Journal (REEN)	4
ACM Transactions of Software Engineering and Management (TOSEM)	2
Empirical Software Engineering Journal (EMSE)	9

information model (TIM) which describes the artifacts and trace link types needed for safety evidence traceability. Rempel et al. [245] provide an approach to parse safety standards in order to identify which trace link types are needed in order to fulfill that standard and check the suggested trace link types against the trace links maintained in the company to determine if they are compliant.

In addition to the mini mapping study, previous systematic literature reviews such as [61], [25] and [257] on traceability as well as overview papers on traceability research such as [34] also support our observations. For instance, [82], [61] and [8] all report that practitioners lack knowledge and guidance on traceability management and further empirical studies that yield guidelines for practitioners are needed.

We discuss the few studies that exist on the overall design of traceability strategies in Section 7.2.1. Additionally, industrial case studies which report on the introduction of traceability are relevant for our research since they provide lessons learned and experiences from industry on how to plan for and introduce traceability. These case studies are discussed in Section 7.2.2.

7.2.1 Frameworks for designing traceability strategies

The study by Rempel et al. [246] discusses the suitability of explicit traceability strategies for different companies and different projects. The authors study existing traceability strategies and development processes in 17 companies and show that there is a mismatch between existing strategies, the development processes, and the project-specific traceability goals. These findings emphasize the need to systematically define a traceability strategy based on the current development process and traceability needs before implementation. The authors therefore propose a framework to investigate the suitability of already existing traceability strategies. TracIMo uses the steps provided in this framework to understand an organisation's goals and existing process. TracIMo then extends

the framework with steps that make it possible to design, deploy and evaluate traceability strategies.

Similarly, the book by Gotel et al. [10] contains a chapter that describes a traceability process model. This model consists of three main activities: planning and managing the traceability strategy; creating and maintaining trace links; and finally using them. The activity for planning and managing of traceability strategy ensures that the traceability strategy is designed according to the needs of the specific project or organization. TracIMo assumes similar concepts and there is some overlap with the steps in TracIMo. However, TracIMo's activities are more detailed and concrete and include specific steps, roles, and work products that are involved in defining a traceability strategy.

Closely related to our study is research on tailoring traceability to specific domains. Dömges and Pohl [18] define a framework for designing project-specific traceability strategies. Their work investigates existing tools and gives guidelines on how to design a traceability management tool that supports definition of project-specific traceability strategies. Their framework is similar to ours as it stresses the need to investigate which traceability strategy is suitable for which project. However, the framework is tool-oriented, defined on an abstract level (without concrete steps of how each activity should be conducted), and does not discuss how to measure and evaluate the designed strategy.

Espinoza and Garbajosa [56] propose a traceability metamodel for the definition of traceability strategies. The authors report that in order to design traceability strategies that are not specific to a development process, it is important that traceability tools support the definition of custom trace links (e.g., `satisfied_by`), user roles (e.g., tester), and definition of linkage rules (e.g., when a requirement and a test should be linked with a `tested_by` trace link). The proposed traceability information model (TIM), i.e., a model describing artifact types and permissible trace link types in a development environment, can be used to define company-specific traceability strategies. However, their work is geared towards defining the traceability information model but not the process. It does not provide details on aspects such as metrics to evaluate the process and tool selection.

Additionally, Mäder and Gotel [258] describe steps for defining project specific traceability which consists of the first three steps in TracIMo but do not go as far as tool selection and evaluation of the traceability strategy designed.

7.2.2 Case studies on introducing traceability

Arkley and Riddle [17] describe tailoring traceability to meet the business needs of a company. Based on an investigation of why the company needed traceability, they derived a successful traceability strategy. This work is similar to ours as it stresses the importance of understanding why specific projects or companies require traceability before introducing any traceability strategy. However, the tailoring approach is not systematized and therefore the steps are not easily transferable. Asuncion et al. [58] conducted a case study on designing and implementing an end-to-end traceability management tool. From the lessons learned in the case study, they provide guidelines on how to establish traceability. While the guidelines are useful, they are

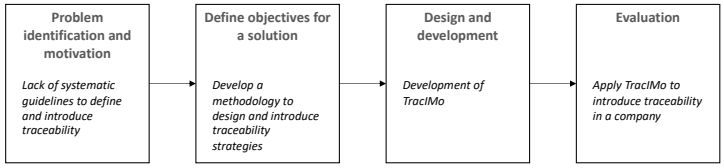


Figure 7.1: The design science process we followed for this contribution. Adapted from [259].

more tool-oriented than process-oriented. Similarly, Kirova et al. [32] report their experiences of implementing an automated traceability environment for a mobile phone company. The study provides guidelines that practitioners should consider when introducing traceability. While some of these guidelines overlap with the steps proposed in our methodology, we give concrete details on how to instantiate these steps. Panis [248], describe a successful implementation of traceability at Teradyne. The author describes the trace link types maintained as well as traceability benefits such as identifying unimplemented requirements, identifying the rationale of requirements during implementation and so on. The study gives recommendations for success such as making sure traceability is available everyday tasks of developers and not a separate report. Stål et al. [255] report on a successful industry developed traceability solution to support continuous integration and delivery at Ericsson. In this study, the authors report how the solution maps to the needs of the company by first eliciting these needs through interviews with practitioners. This study shows how to align the needs of the company to the solution as well as how to evaluate the traceability solution.

In summary, there are very few studies on the introduction of traceability which leaves practitioners with a knowledge gap on how to establish traceability in software development projects. *Our study aims to address this gap by providing TracIMo, a methodology to introduce traceability with concrete, actionable steps and activities. We also report on practical insights into how it was used to establish traceability in a company.*

7.3 Research Method

In this study, we used design science [68] as our research method. Design science allows the researcher to systematically investigate a problem, create artifacts to solve the problem, and evaluate how the artifacts solve the problem in a certain context [68]. The aim of design science is to solve a real world problem through designing innovative artifacts. We used design science because the problem we study (how to systematically introduce traceability) is a practical problem and the goal of our research was to design and evaluate an artifact (a methodology for how to systematically introduce traceability). We followed the design science activities described by Peffers et al. [259]: 1) problem identification and motivation, 2) Define the objectives for a solution, 3) Design and Development, and 4) Evaluation. These activities are described in the next subsections. Figure 7.1 shows the design science research methodology process of our study.

7.3.1 Problem identification and motivation

This first step in design science is aimed at understanding the problem. In our case, the business analyst (BA) of a company reached out to us with a traceability problem. This was followed by emails and phone conversations where two researchers collected data to understand what the problem was. The BA explained that at that point in time, the organisation's development process lacked traceability and, as a result, manual impact analysis was time consuming and error prone. If a change was required all artifacts related to the change needed to be manually identified. Additionally, in many cases development artifacts were out of sync because the change set identified during a change was incomplete. The company therefore wanted to introduce traceability to deal with this problem but did not have the necessary expertise for such an endeavour. From a research perspective, this was a valid problem that is not only relevant for this particular company but also for many others as reported e.g., in Mäder et al. [123] and Maro et al. [82]. Design and introduction of a traceability strategy is a challenging task for organizations because there are no systematic guidelines for practitioners on how to introduce traceability [85] (see also Section 7.2). As such, practitioners can end up managing traceability in an ad hoc manner that where created trace links are not used as they do not support the activities in the development lifecycle.

7.3.2 Define the objectives for a solution

From previous work on software traceability (e.g., [18,26,48,82,83,260] discussed in Section 7.2) we know that ad hoc definition of a traceability strategy is bound to fail since it leads to wasted effort in creating and maintaining trace links which are not used or underused. A traceability strategy needs to be systematically designed in order to reap benefits. Currently, practitioners struggle with defining tailored traceability strategies due to the many aspects involved in making the strategy a success, e.g., making sure the strategy captures stakeholders' needs, is aligned with the development process and supported with proper tools [10]. Our objective is therefore to provide an instrument to help practitioners with the definition and evaluation of traceability strategies for their specific contexts. We achieve this by defining a methodology for systematically designing and introducing traceability strategies.

7.3.3 Design and development

In this step, we developed a methodology to systematically introduce traceability, TracIMo. Before designing TracIMo, we went through literature on traceability to find studies that provide theories and frameworks on how to introduce traceability. To the best of our knowledge we are aware of the related work described in Section 7.2. None of the papers mentioned there provides a methodology with concrete steps and activities for defining a traceability strategy that the company could use to design their traceability strategy. However, various papers, e.g., [83] and [19], call for the need of more guidelines for practitioners on designing and implementing traceability strategies.

We (the researchers together with the BA at the company) used these existing studies as a foundation to derive a more fine-grained and concrete

methodology to introduce traceability at the company. The aim was to make sure that the method captures all tasks necessary to not only design but also introduce and evaluate traceability in an industrial setting. We used the assessment framework by Rempel et al. [245] as a starting point and analyzed which steps would be needed to allow the design, deployment and measurement of a traceability strategy in a systematic manner. This led to addition and modification of some steps to the Rempel framework. For instance, the assessment steps were extended to also include the definition of metrics that would allow tracking the success of the designed traceability strategy in a project team or an organisation over a longer period of time. Importantly, we added steps for adapting tools, deploying the strategy, and measuring its effects in the organisation.

The design of the methodology was done in an iterative manner through brainstorming sessions between the researchers and between the researchers and the BA at the company. The researchers created the first version of the methodology, discussed and improved it in several brainstorming sessions and once a stable version evolved, it was shared with the BA of the company to gauge its feasibility and facilitate further improvements. This was done over a period of two months. The resulting methodology was then used to design, introduce and evaluate a traceability strategy at the company.

7.3.4 Evaluation

In this step we evaluate the applicability of our design artifact (TracIMo).This was done by using TracIMo to design a traceability strategy for the company we collaborated with. The next subsections describe our case, data collection procedure, and how we analyzed the collected data.

7.3.4.1 The case and context

We applied TracIMo to an agile development team of a company in the finance domain to introduce traceability. The company is a digital mortgage advice company located in Amsterdam, whose main business is to provide customer-tailored advice about mortgage products and connect customers to money lenders. The company develops a web-application where customers can register, select mortgages, provide documentation for eligibility, and book appointments with mortgage advisers. The company is small, where the IT department consists of around 14 employees. The main problem for the company was the inability to perform impact analysis when a change requests comes in. From time to time, the company receives changes from the central federal bank on how mortgages should be issued including how the rates should be calculated. The company translates the change request into requirements which are then broken down into tasks and assigned to developers for implementation in the system. Due to lack of traceability, the impact analysis of the new requirements is performed manually and therefore time-consuming and error prone. We used the steps defined in TracIMo to design a traceability strategy that would tackle this challenge in the company. The development team in the IT department was our unit of analysis.

7.3.4.2 Data collection

We the researchers used TracIMo to define and deploy a traceability strategy for the company. This was done by performing each step as prescribed in TracIMo in several iterations and in close contact with the company. After the traceability strategy was defined, we deployed it at the company and collected data to understand its feasibility, again as prescribed by TracIMo. Data was collected for evaluation in three iterations. The first iteration was conducted in the same week as the designed methodology was deployed. We conducted one semi-structured interview with the product owner to get his opinion and feedback on how to improve the strategy. We interviewed the PO because he was not involved in the initial design of the strategy to get his opinion on how the strategy works and fits in their development activities. The interview was recorded and transcribed for analysis. We also conducted one focus group meeting with the lead developer and the BA to elicit their feedback. During the focus group, the researchers took notes which were later used in the analysis. The second iteration was conducted after two weeks. This made sure that the interviewed stakeholders had time to work with the traceability strategy. We conducted one interview with the BA via Skype to understand how the traceability strategy works out for them. The third iteration was conducted after five months where we interviewed the BA and one developer for more feedback on how the strategy worked. All interviews were recorded and transcribed. We also collected data from the bug tracking system used by the company. For instance, we collected the number of closed tickets per sprint since the measurement plan required this data to evaluate the different metrics.

7.3.4.3 Data analysis

We used thematic coding to analyze the transcribed data. The two researchers first coded one interview separately and later held a coding workshop to discuss the codes they came up with and harmonize them. The codes were inspired by TracIMo and our research question. For example, we had a code specifically for the traceability process and for challenges. The harmonized codes were then used for the rest of the interview transcripts. In total we coded four interviews. We also coded the notes that were taken in the focus group meeting using the same codes. The data collected from the bug tracking system was analysed using the metrics defined using TracIMo.

7.4 TracIMo: A Methodology to Introduce Traceability

In this section, we describe TracIMo, the Traceability Introduction Methodology, which can be used to establish traceability strategies in companies. The methodology, depicted in Fig. 7.2, consists of ten steps which are split into two phases. Since TracIMo reuses and extends parts of Rempel et al.'s traceability assessment methodology [246], Fig. 7.2 indicates whether each step was reused as is, modified, or added. One step was reused, four steps were enhanced and five steps are added. We also describe the purpose, the inputs and outputs, as well as the activities for each step.

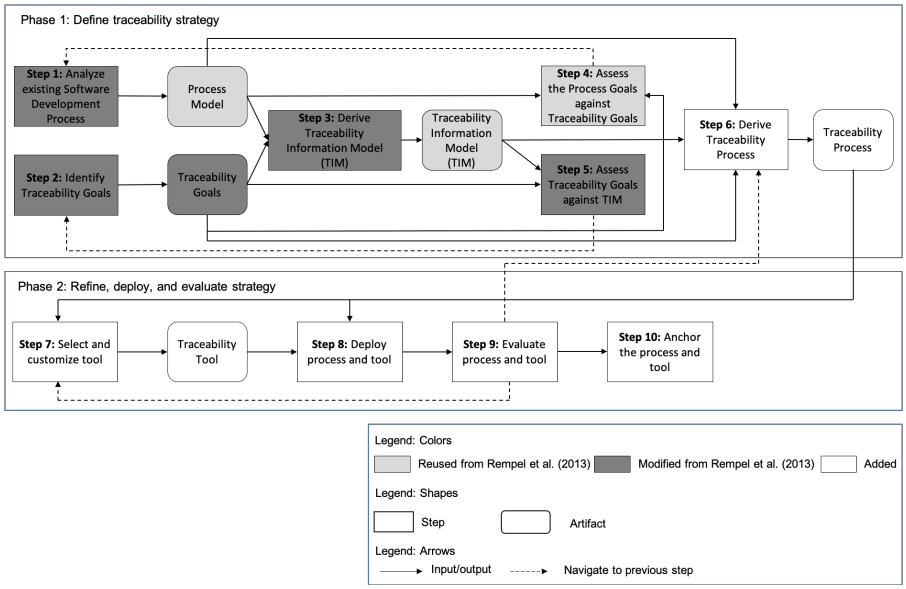


Figure 7.2: Schematic overview of TracIMo, indicating which steps have been reused or modified from Rempel et al. [246] and which were added. The dashed lines represent going back to previous steps for refinement since TracIMo is iterative.

7.4.1 Phase 1: Define traceability strategy

The aim of TracIMo's first phase is to understand the issues and the goals of the company and prepare a suggestion for a suitable traceability strategy.

7.4.1.1 Steps 1 and 2 – Analyse Development Process and Traceability Goals

The purpose of Step 1 is to understand the development process of the company while the purpose of Step 2 is to identify traceability goals. Since these steps use the same data as their foundation, they are presented together.

Activities The main activities in these steps are:

[a] *Collect data on the development process and traceability goals.* This data can be collected through interviewing members of the development team, observing the development team or studying process documentation that describes the development process and traceability needs, or a combination of these data collection techniques. This should be done in close collaboration with the company and include different roles e.g., developers and analysts in order to get the full picture of the development process and traceability needs. For interviews, we propose an interview guide that we have created and made available as part one of the online appendix[1]. While observations and document analysis are good ways to

[1] `https://tinyurl.com/y3n96ldq`

determine the status quo within the organisation, interviews are the main source of information about practiced process and traceability goals.

[b] *Analysis of the data to derive process goals and traceability goals.* This is achieved by going through the data collected either through interviews, documentation or observations. Thematic coding can be used to analyse the transcribed interviews, observation notes, or process documentation. The coded data can then be used to derive a conceptual model of the process which can be modelled using a language like SPEM[2] or Essence[3] or a non-formal format that shows the flow of information between activities. This information is later used in Step 6 to derive the traceability process and ensure that the traceability process is aligned with the development process.

Coded data from the interviews can be used to derive process and traceability goals. Process goals state what the organisation wants to achieve with the different activities in their development process. For instance, in the requirements engineering activity, one of the process goals could be to effectively identify which requirements have already been validated from the requirements engineer's perspective. This information is later important to identify conflicts with the traceability goals and also identify traceability goals that do not support any process goal. The traceability goals in turn describe what the organisation would like to achieve with the introduction of traceability. Both types of goals should also include a rationale that describes the goal further and clearly states why it is important for the organisation.

As an addition to the Rempel et al. framework, TracIMo uses the Goal/Question/Metric (GQM) approach [84] to achieve a standardised format for the goals. They follow the format purpose, issue, object, viewpoint. Purpose is a verb such as "increase", "decrease", or "limit", the issue describes the problem being addressed such as "correctness" or "speed", the object defines what the goal pertains to such as "effort estimations" or "test coverage", and the viewpoint is one of the roles such as "developer", "product owner", or "customer". An example of a goal defined using GQM could be "increase the correctness of identifying change sets for a given requirement, from the developer's point of view."

[c] *Derive metrics for traceability goals.* The GQM approach is also used to define questions and metrics that allow understanding if a goal has been achieved and measuring the success of the derived strategy. For each traceability goal, questions are defined whose answers help understand if the goal has been achieved. For each question, metrics are defined that provide quantitative and qualitative evidence to answer the questions. For instance, the example traceability goal "increase the correctness of identifying change sets for a given requirement, from the developer's point of view" could be associated with the metric "fraction of the number of artifacts in the change set identified during change impact analysis using trace links and the actual number of artifacts changed". If this

[2]https://www.omg.org/spec/SPEM/About-SPEM/
[3]https://www.omg.org/spec/Essence/About-Essence/

metric is close to one, then the trace links fulfill the goal of identifying a correct change set. Additionally, a measurement plan is required for each metric that defines how and when to collect the information. For instance, a measurement plan can state that measurements are taken at the end of each sprint, some after two sprints and some at the end of the project. The measurement plan should also include details how the data for the metrics will be collected, who will be responsible for taking these measurements and how the measurements will be communicated.

[d] *Create exemplary traceability scenarios.* Another addition to Rempel et al. is that TracIMo recommends the definition of scenarios. These scenarios are concrete examples for how trace links are going to be used. Scenarios are a helpful tool in the evaluation of the goals and the traceability information model. We recommend to define a small set of typical exemplary artifacts as they would be created during development and describe how these artifacts should be related to each other and to which purpose. Each traceability goal can be associated with one or several scenarios. A good starting point for definition of traceability usage scenarios is the work by Bouillon et al. [4], who conducted a survey with 56 traceability practitioners and identified a list of 29 traceability usage scenarios relevant for practitioners.

Output The outputs of step 1 and 2 are:

[a] a conceptual model of the process, including roles, activities, artefacts, and tools that are used in the development process;

[b] the process goals along with their rationales;

[c] the traceability goals along with their rationales;

[d] traceability metrics and a measurement plan; and

[e] exemplary traceability scenarios.

7.4.1.2 Step 3 – Derive Traceability Information Model

The purpose of this step is to define a company-specific *Traceability Information Model (TIM)* that adheres to the traceability needs of the company. A TIM captures the semantics of the trace links and provides the structure of the links. It defines which artifact types can be linked to each other and which cardinalities and directions the links have. Depending on the traceability goals, a link can also carry additional meta-data, such as when it was created or who created it.

Input The inputs to this step are:

[a] the process model from Step 1;

[b] the traceability goals from Step 2; and

[c] the traceability scenarios from Step 2.

Activities To derive the TIM, the following activities should be conducted:

[a] *Identify trace link types and traceable artifacts from the traceability goals and the process model.* The traceability goals inform which link types are needed as well as the semantics they have to carry, while the process model informs which traceable artifacts are available in the development process. For instance, system requirements and software requirements are produced in the requirements elicitation process and a relevant traceability goal could be *"to understand how system requirements are broken down into software requirements from the point of view of the business analyst"*. Based on this traceability goal and the associated traceability scenarios, we derive a trace link type that connects system requirements to the software requirements it generates. The use of traceability scenarios constitutes an extension in comparison to Rempel et al. [246]. Additionally, if one system requirement can have many associated software requirements, but one software requirement only has one parent system requirement, the link cardinality can be defined as one to many (1..*), for this link type. When identifying traceable artifacts, it is important to check that these artifacts can be uniquely identified in the development process, as this is a pre-requisite for traceability implementation. In case artifacts cannot be uniquely identified, unique naming schemes should be introduced. To derive the complete TIM, all traceability goals should be analyzed in this way.

[b] *Represent the link types in a model.* After all the link types have been identified, they should be represented in a model for easy presentation. A common way to represent a TIM is to use UML class diagrams or a similar formalism. Textual representation is also possible, but for easy visibility during discussions on the TIM, TracIMo recommends a graphical representation. Figure 7.3 shows an example of a TIM with one link type called "`generates`" that connects system requirements to software requirements. The example also shows that one system requirement can generate many software requirements.

[c] *Identify duplicate trace paths.* Once the TIM is developed it should be checked for different trace paths that link the same elements and have the same semantics. The traceability scenarios can again support this task since it is possible to apply the created TIM to the selected artifacts and see how they would be connected. Duplicates should be removed from the TIM as they will add to the effort of creating and maintaining links but do not yield benefits.

Output The output of step 3 is:

- the traceability information model (TIM).

7.4.1.3 Step 4: Assess Process Goals against Traceability Goals

The purpose of Step 4 is to assure that process goals and traceability goals are compatible and achievable. In particular, it is necessary to evaluate if all

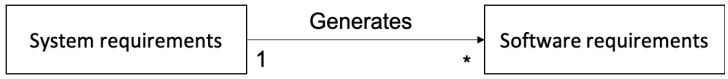


Figure 7.3: An example of a simple TIM with one trace link type (generates) which connects system requirements to software requirements.

process goals that require traceability are covered by at least one traceability goal.

Input The required inputs are:

[a] the process goals from Step 1;

[b] the traceability goals from Step 2;

Activities To assess the traceability goals w.r.t. the process goals, the following activities should be performed:

[a] *Identify process goals that require traceability.* Each process goal has to be evaluated to understand how traceability can support it. This is supported by the rationales of the process goals. For instance, a process goal that is related to translating requirements into a high-level system model has a relation to traceability since the elements in the system model should be traceable to the requirements they address. This step will yield a list of process goals that have to be aligned with traceability goals.

[b] *Match traceability goals to specific process goals.* The list of relevant process goals is then matched to the traceability goals to ensure that there is alignment between what the organisation aims to achieve with the development process and what it expects from a traceability strategy. This is done by going through all the process goals that require traceability identified previously, and checking if each of these goals have at least one corresponding traceability goal that supports the process goal. This step can lead to a refinement of the goals or even to revisiting the goals internally to determine which are of highest priority in case some goals cannot be fulfilled.

Output The outputs of this step are:

[a] an assessment report of the process goals against the traceability goals, represented as a table relating the process goals and the corresponding traceability goals;

[b] a list of refined process and/or traceability goals (optional).

7.4.1.4 Step 5: Assess Traceability Goals against TIM

The purpose of Step 5 is to assure that the TIM's structure supports the traceability goals. In particular, it is necessary to evaluate if the TIM supports the storage and analysis of all necessary information to achieve them.

Input The required inputs are:

[a] the traceability goals from Step 2;

[b] the traceability scenarios from Step 2; and

[c] the TIM from Step 3.

Activities To assess the TIM w.r.t. the traceability goals, the following steps should be taken:

[a] *Identify trace link types associated with each traceability goal.* Traceability goals often imply that certain artifacts should be traceable to each other. A traceability goal about being able to identify missing test cases, e.g., implies that test cases are connected to requirements, to design models, or to source code. Such information can also be derived from the rationales of the goals.

[b] *Check that all required link types are represented in the TIM.* In this step, it is not only important to check that the TIM contains all required links, but also that the TIM has no links that are not connected to any traceability goals. In addition to Rempel et al., example trace links for specific traceability scenarios from Step 2 should be created to determine if the TIM's expressiveness is sufficient. In case of misalignment, the goals and the TIM are revisited iteratively until converging towards a solution. This provides an early evaluation of the suitability of the TIM.

Note that these assessment steps are iterative and can lead to changes in the traceability goals, the TIM as well as the process goals.

Output The outputs of this step are:

[a] an assessment report of the traceability goals against the TIM, represented as a table relating the traceability goals to a description of how the TIM supports them; and

[b] a list of exemplary trace links created for specific traceability scenarios.

7.4.1.5 Step 6: Derive traceability process

The purpose of this step is to define an explicit traceability process. The traceability process defines the traceability activities (e.g., creation, maintenance and usage of trace links), as well as the roles responsible for each of the activities. It also defines a workflow of how and where in the development process, trace links will be created and maintained as artifacts evolve. If any automation will be used to create links or enforce the traceability workflow, this also needs to be defined in the traceability process. Finally, the traceability process describes how and when to use established trace links.

Input The inputs to this step are:

[a] the process model from Step 1;

[b] the traceability goals and associated metrics from Step 2;

[c] the traceability scenarios from Step 2;

[d] the TIM from Step 3.

Activities To derive the traceability process the following activities are conducted:

[a] *Identify when trace links will be created.* The current process model is used as a foundation to define the process stages in which links will be created. In order to understand when this should happen, the traceability goals, their associated traceability scenarios, and the TIM can be used. For instance, a scenario could show that links between requirements and test cases should be created during requirements analysis. The concrete link type is defined by the TIM. Existing activities in the process model can be extended or new activities can be created.

[b] *Identify which roles will create trace links.* The roles responsible for creating certain link types are partially prescribed by the activity in the process model. If a link is created during requirements analysis, e.g., the roles involved in this activity are candidates to take on the responsibility for the creation of the link. However, the analysis based on the traceability scenarios and traceability goals may show that additional roles need to be involved. The viewpoint that is part of the traceability goal can be a helpful pointer here.

[c] *Identify when trace links will be updated or deleted.* In order to avoid that the trace model becomes stale, trace links need to be updated or even deleted. This can, again, happen during existing activities in the process or during newly defined activities, if necessary. It is possible that several activities are extended to update or delete links.

[d] *Identify which roles will update or delete the trace links.* Likewise, who is responsible for the update or deletion of trace links needs to be defined in the process model.

[e] *Identify when and how trace links will be used.* Using the traceability goals and the traceability scenarios, the activities in which the traceability information is used are defined. At this stage, it is also important to describe how and when the links are used (e.g., to find dependencies or identify missing tests).

[f] *Identify which roles will use the trace links.* Finally, which roles are going to use the trace links is defined. The viewpoint in the traceability goal can give insight into this. It is important to note that the respective roles need access to the trace model and the artifacts the trace links connect in order to use them effectively.

[g] *Integrate measurement plans.* The definition of the traceability goals also included metrics and associated measurement plans. Collecting the data needed and recording the measurements should be included as explicit activities in the traceability process along with responsible roles and a defined way to access the information.

The outcome of each activity should be captured in a traceability process model. Among other things, it contains the link types to be created, how they will be created, who will create them and how the links will be updated. The process can be documented in different ways based on the level of formality required. If a formal description is necessary, e.g., to integrate it into an existing formal process description, modelling languages such as SPEM or Essence can be used. On the more informal end of the spectrum, wiki entries or even just informal communication within the team can be used. However, TracIMo recommends to document the traceability processes in written form in order to be able to revisit and evolve it. The aforementioned modelling languages also provide hints on what should be documented. Activities, e.g., should include a purpose, input and output, the role responsible, and the concrete steps to be taken.

Output The output of this step is:

- a traceability process model.

7.4.2 Phase 2: Refine, deploy, and evaluate strategy

The aim of the second phase of TracIMois to deploy the traceability strategy and evaluate its effectiveness.

7.4.2.1 Step 7: Select and Customize tool

Once the conceptual traceability strategy is created, the company needs to think about tool support for the different activities that need to be carried out. These activities include creation, maintenance and use of trace links. If tool support does not already exist, a traceability management tool needs to be selected and customized to support the different traceability activities.

Input The input to this step is:

[a] the process model defined in Step 1;

[b] the TIM defined in Step 3; and

[c] the traceability process defined in Step 6.

Activities The following activities are conducted in this step:

[a] *Identify tool requirements from the traceability process.* The TIM and the process provide information about which links have to be created and which artifacts need to be supported. The need to link requirements stored in spreadsheets to design models in UML, e.g., means that the

traceability tool has to support tracing to and from spreadsheets, and to and from UML models. Additionally, the tool needs to support granularity and different link directions if so specified by the TIM. The process model also provides other information such as the existing tool chain. Additional requirements might be elicited here, e.g., if the solution can be commercial or has to be available without license fees.

[b] *Analyze existing traceability tools and select tool based on derived tool requirements.* Scientific literature provides some guidance on traceability tool selection that can be used to facilitate the process. Rempel et al. [48], e.g., gives an overview of steps to elicit tool requirements and important factors to consider. Additionally, Gotel and Mäder [16] provide characteristics that can be used to compare different traceability tools. The latest work is a study by SteghÃűfer [261] which defines categories that can be used to assess trace links aimed at helping practitioners identify which tools are suitable for their needs. The paper uses factors and guidelines defined in Maro et al. [26] to define concrete traceability tool characteristics and provides an evaluation of 23 existing traceability tools based on these characteristics. We recommend that a systematic assessment of the tools is done using the categorisation defined by SteghÃűfer [261] or Rempel et al. [48]. However, the characteristics or criteria from these studies should only be used as a starting point and the systematic assessment should focus on the traceability tool requirements from the company which are inferred from the existing development process, existing tool chain, existing skills and knowledge, as well as the traceability goals and the TIM. In case there is not tool to support the traceability needs of the company, the company can develop an in-house solution. It should be noted that in some cases, more than one tool is needed to satisfy the traceability goals of the company.

[c] *Customize selected tool.* Since every company has unique requirements when it comes to traceability, it is common that the selected traceability tool needs to be customized to fit the company needs. At the very least the tool needs to use the TIM defined in previous steps. Additional customisations can, e.g., include collection of data for use in the metrics. It is important to ensure that the selected tool can be customized in a reasonable time frame and cost.

Output The outputs of this step are:

[a] an assessment report of existing traceability tools and reasons for selecting the tool which can be used to justify how the tool was selected and how it fits the company needs;

[b] a customized traceability tool or an off-the-shelf traceability tool or an in-house developed tool.

7.4.2.2 Step 8: Deployment of the designed traceability strategy

The purpose of this step is to deploy the traceability strategy, which consists of the traceability process and customized traceability tool, at the organisation.

TracIMo recommends to deploy the process incrementally, i.e., one project at a time.

Input The inputs of this step are:

[a] the traceability process from Step 6; and

[b] the customized traceability tool from Step 7.

Activities The following activities are required for the deployment:

[a] *Create a deployment schedule.* This schedule defines when the tool will be installed at the company, when training takes place, when the traceability tasks will start and who will be responsible for each process. To ensure a successful deployment, it should be scheduled explicitly. In agile environments, the roll-out can e.g., be included as a task in sprint planning or the velocity can be lowered for the sprints in which the new activities are introduced.

[b] *Create baseline measurements.* In order to measure the effectiveness of the traceability strategy, it is important to create a baseline against which the new process can be compared. For this purpose, initial measurements according to the measurement plan for the metrics associated with the traceability goals should be taken now. This also ensures that the necessary steps to collect measurements used to evaluate the metrics are in place.

[c] *Inform all involved stakeholders.* All involved stakeholders should be informed of the process, how it is going to affect their work and what is expected of them. For instance, it is important to make the roles responsible for each task in the traceability process aware of their new duties. It is also important to ensure that those who create the links know whom they create them for. This can be done by distributing the process documentation created in Step 6 as well as the deployment plan. Personal discussions with the stakeholders can ensure buy-in and alleviate anxiety associated with the changes.

[d] *Train involved stakeholders.* Before deployment, all stakeholders should participate in training activities such as workshops that demonstrate the new activities and allow the responsible roles to develop the skills to perform them. These workshops can also be used to teach the relevant tools.

[e] *Integrate the traceability tool into the development tool-chain.* The traceability tool has to be included into the development tool-chain and installed on the machines of all stakeholders that produce or consume trace links before the process is rolled out.

[f] *Roll-out the process.* Once training is complete and all necessary tools are in place, the traceability process can be rolled out. Again, this roll-out should be scheduled accordingly and can have an impact on the velocity in the first sprints after roll-out since additional time might be required

for on-the-spot training or due to issues that occur when the traceability process is applied in practice for the first time.

Output The outputs of this step are:

[a] a deployment plan describing the concrete steps and their timing to introduce traceability tools, practices, and training;

[b] baseline measurements before the roll-out of the traceability process; and

[c] a deployed traceability strategy used by the involved stakeholders.

7.4.2.3 Step 9: Evaluation

The purpose of this step is to evaluate the deployed process in order to find out if the traceability goals are achieved and identify areas of improvement.

Input The inputs to this step are:

[a] baseline measurements before the roll-out of the traceability process from Step 8; and

[b] traceability goals and associated metrics from Step 2.

Activities While there are different ways to evaluate the deployed process, TracIMo recommends the following evaluation activities:

[a] *Immediate evaluation of the strategy during the deployment period.* The evaluation is performed by collecting data according to the measurement plans and analysing it using the metrics defined in Step 2. It is also helpful to observe how the stakeholders work with the process and tool and also discuss the process with the stakeholders. The discussions could be informal meetings, focus groups or structured interviews depending on the company and availability of the stakeholders. Since unanticipated challenges can occur, lessons learned from the deployment should quickly be taken up and the tool and process improved as necessary.

[b] *Long term evaluation of the strategy.* The metrics defined in Step 2 can also be used to monitor the success of the strategy over a longer period of time, in particular in terms of improvement over the baseline. For instance, after the first three months, the measurements can be analyzed to identify areas of improvement and traceability goals that are not fulfilled by the existing traceability strategy. Additionally, a qualitative evaluation of the established strategy should be conducted. The involved stakeholders can be interviewed for their views in the strategy in order to elicit areas of improvement.

Output The output of this step is an evaluation report that contains details on how well the traceability strategy works and which areas need improvement. The evaluation report also contains the measurements that were taken to show to what extent the goals have been achieved with the deployed strategy as well as lessons learned and recommendations for future improvement. Depending on the organizations' requirements, this report can be a formal report or informal documentation stored as, e.g., a wiki page.

7.4.2.4 Step 10: Anchor process and tool

The purpose of this step is to anchor the new process and tool within the team and the organisation. This step requires that the traceability strategy is deployed and used in the organisation.

Activities TracIMo recommends the following activities to ensure that the deployed process is anchored:

[a] *Continuously educate developers and stakeholders.* Both current and new employees need to be educated about traceability, its benefits, and the necessary steps to incorporate it into the development process continuously.

[b] *Integrate traceability in reviews.* To ensure that traceability activities are performed, their outcome can be included in code reviews and sprint reviews or other opportunities for feedback. In code reviews, the guidelines can, e.g., state that new test cases need to be traced to the original requirement for the review to pass. Likewise, in sprint reviews the trace model can be reviewed to find missing links or links that need to be updated or deleted.

[c] *Include traceability metrics in dashboards.* Many development teams use dashboards (see, e.g., [262]) to visualise the current state of the product being developed. Some metrics about traceability can be evaluated automatically and integrated into these dashboards to provide a view on the quality and number of trace links and how they support the team. Indirect metrics (e.g., accuracy of estimates) can also be visualised this way to incentivise stakeholders to stick to traceability practices.

Such steps often require a more formalised definition of the traceability process. If this was not done in Step 6, the traceability process description should be revisited. At this point, going through another iteration of TracIMo can also be useful to establish additional process and traceability goals and refine the process to accommodate more teams.

Output The outputs of this step are:

[a] updated training material for the development process and the traceability strategy;

[b] guidelines for including traceability in reviews; and

[c] automated measurement of relevant data for traceability metrics and inclusion in dashboards.

7.5 Application of TracIMo in a company

To evaluate TracIMo, we applied it in a company in the finance domain. As previously mentioned, the company develops a web-application which customers use to select mortgages, provide documentation for eligibility, and book appointments with mortgage advisers.

This section describes how TracIMo was applied to introduce traceability in the company. Since some of the steps (e.g., Step 1 and Step 2) require the same data, they can be carried out in parallel.

7.5.1 Step 1 and 2: Analyze existing process and identify traceability goals

To understand the development process and the traceability goals for the company, we conducted two interviews, one with the business analyst and one with the lead developer. The interviews were conducted via Skype and each interview lasted around one hour. The interview guide we used is described as part 1 of our interview guide document available online[4]. Both interviews were recorded, transcribed and analyzed. We used the thematic coding approach [263] on the transcribed data. Examples of the codes we used are *"process goal", "traceability goal", "traceable artifact", "traceability challenge", and "trace link type"*. These codes are derived from what TracIMo prescribes. Even though we know exactly what we are looking for in the transcripts, the thematic coding approach ensures that we derive this information systematically and therefore avoid missing any needed information. Two researchers then used the data to build the conceptual models of the process and the abstract goals. These were checked with the interview partners for accuracy and correctness. Based on the traceability goals and process goals, we applied GQM to derive potential metrics. For example, in an interview with the lead developer he said:

"*Sometimes we underestimate tickets because we forget about some parts of the system which should be touched by the changes and that's a problem.*"

From this quote, we derived the goal *improve the accuracy of effort estimations for tasks, from the lead developer's point of view* which is detailed in Table 7.2. Table 7.2 also shows the rationale of the goal and the metrics derived for evaluating the goal. Further details on the metrics derived for all the traceability goals can be found in our supplementary material describing the case study[5]. The researchers investigated each goal and proposed a number of possible metrics. These metrics were analysed for feasibility together with the BA to see if the data needed to evaluate the metrics is actually available and a subset was selected. Measurement plans for when the measurements should be taken were created for each metric. For instance, the number of deviating tasks is to be measured at the end of each sprint by the BA. For each goal, we also derived traceability scenarios which were later used to asses if the traceability goals are achieved. An example of a scenario defined for goal 3 is also included in Table 7.2.

The results of step 1 and 2 are: 1) the process model, which includes a description of the development process activities and process goals as summarized

[4] `https://tinyurl.com/y3n96ldq`
[5] `https://tinyurl.com/y6dmd8u9`

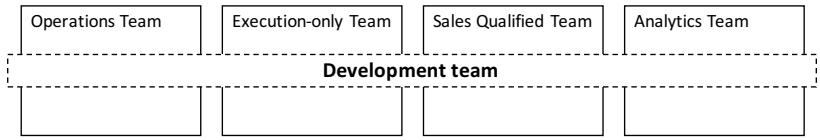


Figure 7.4: Organization structure of the company.

in Table 7.4; and, 2) traceability goals which are summarized in Table 7.5.2. Additionally, the traceability goals include questions, metrics and scenarios as exemplified in Table 7.2. A summary of the development process at the company is given below.

Development process at the case company: The development team uses Scrum and comprises the following roles: product owner, scrum master, developer, and quality engineer (tester). The developer role is refined into back-end developers, front-end developers, UI designers and web designers. In addition, a business analyst is responsible for breaking down high-level requirements into user stories and assuring that the development of the software coincides with business goals and regulatory requirements.

The company is structured into four value teams: operations, execution-only, sales qualified and the analytics team (cf. Figure 7.4). A value team is a group of people with different expertise (development, marketing, operations) that work together to achieve a defined goal. The development team is a horizontal group distributed over four different value teams. Each value team has dedicated developers that implement features to achieve the team's goal. Some of the developers are located abroad and therefore work remotely.

Each value team works autonomously and has a Scrum master who ensures that the Scrum principles are applied correctly and helps team members to address any obstacles. Each value team also has its own product owner who is responsible for defining the team's focus by defining the scope of each sprint. It is possible that some of the development team members are assigned to tasks belonging to different value teams.

Sprints last two weeks. At the beginning of these two weeks, a planning meeting is held to decide which tasks need to be accomplished in the sprint and to assign the tasks to responsible developers. Once a developer is done with a task, they send a pull request. If this is accepted, the changes are deployed to the testing system. Once testing is complete, the feature is released. The sprint ends with a retrospective meeting to reflect on how the sprint went and identify how the process can be improved. Furthermore, at the beginning of each sprint, the product owners from the four value teams gather to coordinate the overall direction and to analyze which steps should be taken next in the roadmap by identifying issues with high business value. Every morning during the sprint, the development team and each value team have separate stand-up meetings.

Table 7.2: Goal/Question/Metric to identify traceability goals and metrics

Goal 3:	Improve the accuracy of effort estimations for tickets from the lead developer's point of view.
Rationale:	One of the main tasks for the lead developer is to estimate the effort a certain implementation task is going to have. This has a major influence on the sprint and on the schedule for the developers since it essentially determines how many tickets the team will tackle during a sprint and how much time they can devote to each ticket. Increasing the accuracy of the effort estimation is therefore a goal. Trace links can support this goal by providing insight into dependencies between artifacts and requirements, and by helping to identify which parts of the code have to be touched for a change. Since an estimation can never be 100% accurate, an additional dimension is how confident the lead developer feels with his estimations. If trace links do in fact support the estimation, the lead developer should become more confident in estimating over time and high confidence estimations should become more accurate at the same time.
Question 1:	How much does the estimated effort differ from the actual effort?
Metrics:	• Average number of tasks per sprint (analysis of Product Backlog/JIRA tickets) • Average number of deviating tasks per sprint (analysis of Product Backlog/JIRA tickets) • Percentage of deviating tasks per sprint (derived) • Initial estimation for each task in story points (analysis of Product Backlog/JIRA tickets) • Updated estimation for each task in story points (analysis of Product Backlog/JIRA tickets) • Average increase/decrease in effort per task (derived) • Number of JIRA comments about effort per task (analysis of JIRA tickets)
Question 2:	How confident is the lead developer in the estimation of tasks?
Metrics:	• Likert scale confidence 1 – not confident at all 5 – very confident per task (Questionnaire with lead developer) • Number of low confidence tasks that required a change (analysis of Product Backlog/JIRA tickets) • Number of high confidence tasks that required a change (analysis of Product Backlog/JIRA tickets)
Scenario:	Given a ticket, it should be possible to identify those parts of the system that are affected by the change in the ticket. By being able to conduct a change impact analysis down to the code, copy, and wireframe level, the lead developer can make better estimations of the tickets. This means that a ticket needs to be linked to requirements, model elements, implementation, tests, copy, wireframes and art designs.

7.5.2 Step 3 and 5: Derive traceability information model and Assess traceability goals against TIM

We analyzed the development process, traceable artifacts and the traceability goals and designed a *Traceability Information Model (TIM)*. Existing traceability practices were taken into account to ensure that the designed TIM supports them. We carried out step 3 and 5 together because of the synergy that exists between the steps. Since the TIM is derived from the traceability goals, we also assessed the TIM with respect to the traceability goals during the creation of the TIM. This ensured that the resulting TIM will fulfill all the traceability goals. This also means that the TIM is created in iterations.

Whether the traceability goals can be achieved or not depends on the expressiveness of the TIM as well as the traceability practices that are put into place. Since we focus on the TIM, the object of the analysis are the artifacts that are connected via trace links and the semantics of these links. In this step, we also used the scenarios defined in Step 2 in the assessment. An example a traceability scenario for traceability goal 3 is shown in Table 7.2. Using this scenario, we assessed if the TIM supports tracing between all relevant artifacts, i.e., from tickets to requirements, model elements, implementation, tests, copies, wireframes and art designs. As can be seen in Figure 7.5, the TIM supports these link types and this scenario. From a ticket, there are direct links to requirements, copies, wireframes and art designs. Additionally, transitive links exist from tickets to model elements, implementation and tests. We also investigate which granularity level the TIM requires, if any and if that is sufficient to fulfill the goal, which is to improve the accuracy of effort estimation. The evidence used in the assessment is mainly the structure of the TIM, e.g., that the right kinds of artifacts are connected to achieve the desired goal. Table 7.5.2 shows all the traceability goals and how the TIM helps to achieve them. The descriptions also provide hints for the practices, e.g., that some trace links can be used for analysis once they are established. After several iterations of feedback from the BA, the TIM shown in Figure 7.5 emerged, which was later deployed in the company. All links in the model are one-to-one (one link can connect exactly two artifacts) and unidirectional as indicated by the directed arrows.

7.5.3 Step 4 Assess process goals against traceability goals

Using the process model and the traceability goals, we assessed the process goals with respect to the traceability goals. This is to ensure that each process goal that requires traceability is covered by at least one traceability goal. This analysis was first done by two researchers who read all the process goals to identify goals that required traceability and identified the matching traceability goal(s) from the list of traceability goals derived from step 2. For instance, one of the process goals is *to improve the understanding of the relationship between code and requirements, from a developer's point of view.* In the assessment, we matched this goal with traceability goal 4, *to increase efficiency of identifying artefacts relevant to a change from BA's point of view.* This is because traceability goal 4 is fulfilled by having trace links from

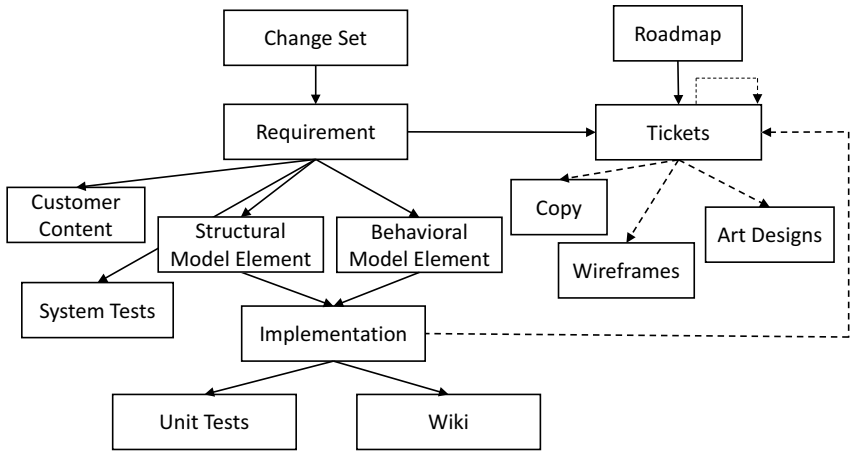


Figure 7.5: Initial traceability information model. Links shown as dotted lines were already captured at the company.

tickets to requirements, requirements to model elements, model elements to implementation and implementation to tests. This makes the artifacts relevant to a change not only visible for the BA but also for the developer. The result of the analysis was then shown to the BA for confirmation and feedback and is summarized in Table 7.4.

Table 7.3: Assessment of Traceability Goals against the TIM

Traceability Goal	How the TIM supports the goal
Goal 1: Increase the awareness of stakeholders about product changes from the BA's point of view.	The change impact analysis enabled by the existence of trace links, allow the business analyst to communicate product changes to the stakeholders and to give an indication which impact they have. For instance, the links show which artifacts are connected to a task and if these artifacts require stakeholders that are not in the development team to be involved. The BA can spot this and inform the appropriate stakeholders.
Goal 2: Improve the visibility of the decision rationale from the development team's perspective.	Links between requirements and tickets allow the developers to go to the requirement(s) associated with a ticket in order to read the rationale of the requirement.
Goal 3: Improve the accuracy of effort estimations for tasks from the Lead Developer's point of view.	Links between the tickets and the model elements allow identifying all aspects of the system that are affected by a change. The transitive links to the implementation and tests indicate the code elements that need to be changed. This change impact analysis improves the overview and should support the development team in estimating the ticket. For instance if a ticket is connected to many complex classes, then it is an indication that the ticket needs more effort.
Goal 4: Increase the efficiency of identifying artifacts relevant to a change from the developers' point of view.	This goal can be achieved due to the same reasoning as for Goal 3.
Goal 5: Increase the efficiency of identifying artifacts relevant to a change from the BA's point of view.	This goal can be achieved due to the same reasoning as for Goal 3.
Goal 6: Improve the visibility of the dependencies of the process steps from the lead developer's point of view.	The process steps correspond to different activities that need to be performed by different stakeholders. For instance, copy needs to be provided before the web page can be programmed. The existence of a trace link between a requirement and copy thus indicates that the step has been done. The developers can therefore plan for tasks based on these dependencies e.g., the task of copy writing will be planned before that of web page development.
Goal 7: Improve the visibility of progress from the PO's point of view.	The TIM makes it easier to track progress since it clearly identifies the elements affected by a change. When comparing with which elements have already been changed (e.g., tests, customer content, models) to which have to be changed, a notion of completeness can be derived. Notably, however, traceability does not help establishing to which degree the different elements have already been completed, just if they have been touched at all.

Table 7.4: Assessment of Process Goals against Traceability Goals (cf. Table 7.5.2). The table shows which process goals map to which traceability goals and how the achievement of the traceability goal supports the achievement of the process goal.

Current Practice	Process Goal	Support by Traceability Goals (TG)
Requirements Engineering: The POs and the BA are responsible for the requirements engineering tasks which are to elicit requirements from the different value teams, document these requirements and make sure they are translated into actionable tasks. The requirements are written in Google Drive spreadsheets so that they can be easily shared. When requirements come from external entities, e.g., regulation boards, they are in PDF format.	**Process Goal 1**: Elicit all requirements from the product owner/BA's point of view. **Process Goal 2**: Allow breakdown of all requirements into actionable tasks from a product owner/BA's point of view. **Process Goal 3**: Improve the identification of related requirements from the product owner/BA's point of view.	Supported by **TG 5** to increase the efficiency of identifying artifacts relevant to change from the BA's point of view. Even though the traceability goal is formulated from the lead developer's point of view, both the BA and PO can use links between tickets and requirements to identify related requirements, e.g., if the requirements are linked to the same ticket.
Software Design: At the beginning of projects, the developers design a high-level overview either on the white board (stored as pictures) or in a UML modelling tool (stored in the corresponding format).	**Process Goal 4**: Improve the understanding of the software requirements from a developer's point of view.	

	Process Goal 5: Allow creation of a high-level design based on the requirements from a developer's point of view.	Supported by **TG 2** to improve the visibility of the design rationale from the development team's perspective, and achieved using a similar reasoning as for Process Goal 3.
Development: The developers work on the tickets assigned to them and produce code. The code is written manually in PHP and stored in git repositories.	**Process Goal 6**: Allow implementing new features from a developer's point of view.	
	Process Goal 7: Allow implementing changes of existing features from a developer's point of view.	
	Process Goal 8: Improve the identification of artifacts that need to change from a developer's point of view.	Supported by **TG 4** to increase the efficiency of identifying artifacts related to a change. Trace links from tickets to other development artifacts show which artifacts are affected by the change described in the ticket.
	Process Goal 9: Improve the understanding of the relationship between code and requirements from a developer's point of view.	Supported by **TG 4** and achieved using a similar reasoning as for Process Goal 8.
	Process Goal 10: Improve the planning process for future changes from a developer's point of view.	Supported by **TG 6** to improve the visibility of the process steps. Trace links allow to determine which parts of the process need to be executed first and therefore plan accordingly. A missing link from model to implementation, e.g., indicates that the code has not yet been written.

Quality Assurance: The developed feature is tested against its requirements to verify that it works correctly. The company has dedicated testers who write tests for implemented features. The tests are stored in git repositories together with the code they test.	**Process Goal 11**: Improve the understanding of requirements from a tester's point of view. **Process Goal 12**: Allow verifying features from a tester's point of view. **Process Goal 13**: Improve the understanding of which artifacts need to be tested after a change is made from a tester's point of view.	Supported by **TG 2** and achieved using a similar reasoning as for Process Goal 3. Supported by **Traceability Goal 4** and achieved using a similar reasoning as for Process Goal 8.
Project Management: This activity is associated with planning development and following up on the progress of development to make sure that features being developed align with the goals of the company.	**Process Goal 14**: Improve the understanding of software requirements from a PO/BA's point of view. **Process Goal 15**: Improve effort estimation of requirements from a PO/BA's point of view. **Process Goal 16**: Allow prioritizing requirements from a PO/BA's point of view. **Process Goal 17**: Improve requirements' progress monitoring from a PO/BA's point of view.	Supported by **TG 2** and achieved using a similar reasoning as for Process Goal 3. Supported by **TG 3**, which is geared towards improving accuracy of effort estimation. PO and BA can use links between the requirements and tickets and to other development artifacts to see how many artifacts will need to be inspected and changed.

7.5.4 Step 6: Derive traceability process

In this step, we defined how trace links were going to be created, maintained and used. The inputs we considered for this step were the process model, the traceability goals, metrics, scenarios and the defined TIM. Since the BA was already responsible for conducting the manual impact analysis, we decided that he should also create the trace links because he knows the system well and was already creating links implicitly in the existing development process. The BA is also responsible for updating the links when artifacts evolve. Due to the difficulty in tracking what has changed manually, it was decided that the BA will need tool support to help maintain the trace links. This requirement was noted and later used when selecting the traceability tool. The end users of the trace links will be the development team, the lead developer, the product owner as well as the BA. Due to the fact that there were no existing links and the systems developed at the company already had a large number of artifacts, the links will be created in a retrospective manner. To reduce the load for the BA, the links will also be created incrementally. For each sprint, the BA will create links to tickets planned for the sprint and make these links available to the developers. This is a lightweight approach for creating links as the BA can focus the effort on the links that yield immediate benefits. Furthermore, links between development artifacts are also created incrementally, e.g., links between model elements and implementation and between implementation and tests. These links can be reused the next time a change involves an artifact that already has trace links.

We used the metrics and measurement plan defined in Step 2 to define a data collection strategy for inclusion in the traceability process. The data from JIRA, e.g., the average number of tickets per sprint, can be automatically obtained from the JIRA system. We agreed that data that had to be elicited from stakeholders, e.g., developers and the product owner, will be collected by the BA.

7.5.5 Step 7: Select and customize tool

To select a suitable tool that will support the defined traceability strategy, we considered the existing development process, the tools used in the company, the TIM and the traceability process defined for the company. The tools used in the development process are depicted in Table 7.6. We also considered additional tool-specific requirements. For instance, it was important to the BA to have tool support in terms of notifications when artifacts evolve, so that he can update the respective trace links. The BA also wanted the developers to have as little change as possible in their tooling. One additional but important requirement from the company was to use an open source tool that would require little customization, because the change was driven by the BA's interest and had no budget for acquiring a commercial tool. This also means that knowledge on how to customize the tool needs to be available. We used the tool categorization defined in [261] where the authors have analyzed 23 existing traceability tools, to select a tool to use. This categorisation evaluates the traceability tools using six main characteristics: 1) information storage, which describes where the tool stores the trace links; 2) level of integration, which describes whether the tool is a holistic tool supporting all software engineering activities or a standalone

Table 7.5: Characteristics used to assess the tool and possible values.

Characteristic	Possible value
Information storage	Centralised, Distributed, Separate Model, Inline
Level of integration	Holistic, Hybrid, Separate
Tool type	Application Lifecycle Management (ALM), Requirements Management, Standalone traceability tool, Integration tool, Special purpose tool, Link recovery tool
Integration context	Tool-chain specific, Framework, Generic
Configuration options	Traceability Information Model, Artifact adapters, Visualisation
Automation	Link generation, Consistency checking, Workflow enforcement

traceability tool; 3) Tool type, which describes if the tool has a specific purpose e.g., requirements management; 4) integration context which describes which other tools the traceability tool can be integrated with; 5) configuration options which describe which parts of the tool are customisable; and 6) automation which describes which trace activities the tool automates. Table 7.5 shows the six characteristics and possible values. We disregarded commercial tools and remained with five tools whose categorisation is shown in Table 7.7.

Based on the tool assessment and the requirements from the company, we selected Eclipse Capra [69] due to the following reasons: 1) it allows the definition of a custom TIM; 2) it can be extended to support additional artifact formats; 3) the visualization can be customized; 4) it supports link maintenance through notifications; and 5) the researchers have the knowledge needed to customize it.

Table 7.6: Development artifacts and tools at the company.

Artifact	**Tool**
Requirements and Copy	Google Drive (Spreadsheets)
Change sets	PDF
Tickets	Jira
Customer content	Media wiki
Models	Papyrus
Code and Tests	Git (PHP code)
Wireframes	Axure (exported as PNG)

The fact that the researchers are familiar with the customisation of the tool was probably the most relevant. Since Eclipse Capra is based on the popular Eclipse IDE[6], it requires the use of this development environment. Additionally, both the BA and the LD had prior experience with using tools based on the Eclipse IDE. However, the company did not use Eclipse at this point in time. This meant that a rather heavy-weight new tool had to be integrated into the

[6]http://www.eclipse.org

development tool-chain. A traceability plug-in for JIRA, e.g., would have had less impact on the tool-chain. However, a plug-in that fulfilled the requirements of the company and allowed achieving the traceability tools was not available. In the end, the willingness of the BA and the lead developer to adopt a new tool that would also allow them to work with the UML models of the software (see below), the fact that Eclipse Capra could be adapted quickly and without additional cost, and that work with Eclipse would be limited to the BA and the lead developer while the rest of the team would only use the results, trumped the concern of introducing a new tool.

In order to support the new traceability process at the company, the traceability management tool Eclipse Capra was customized in three ways: 1) the company-specific TIM from Step 3 was created and incorporated in the tool; 2) two artifact adapters were implemented, one to support linking to and from requirements in Google spreadsheets and one to support linking to and from PHP code; and 3) the visualization of the tool was customized to include direction of the links and to allow filtering based on selected tickets. Overall, this customization took around 3 weeks where one student developer from the university worked on creating the adapter to link to Google spreadsheets and one researcher spend some hours on the rest of the customization.

To use the tool, the BA or the lead developer would import the artifacts into an Eclipse workspace and create the links between them. The links can be shared using a git repository so that it is available to both the BA and the lead developer. After the links are created, the tool can automatically generate a graphical representation of how the artifacts are related to each other. For each ticket, such a graph is uploaded by the BA in the bug tracking system JIRA so that the developers have a clear understanding of the relationships between the different artifacts concerning the ticket. To maintain the links, the tool has a notification feature that shows warnings on artifacts that have changed and are associated with trace links. The responsible person can thus check if the trace links need to be updated as well.

7.5.6 Step 8: Deploy process and tool

The deployment was scheduled to take place during one week. During that week, the two researchers were present full time at the company. The schedule (which can be found on page 18 of the online appendix[7]) was created in collaboration with the BA and the BA communicated this to his team. On the first day, the researchers were introduced at the company and explained the purpose of the visit during the morning stand-up meeting. Since this was communicated to the development team before our arrival, it was brief. Additionally, since only the BA and lead developers were going to be working with the traceability tool and the rest of the developers would only use the images in the JIRA tickets, the development team required no training on the tool but only information on how to use the links.

[7] https://tinyurl.com/y6dmd8u9

Table 7.7: Assessment of traceability tools [261].

Name	License	Type	Information Storage	Level of integration	Integration context	Configuration options	Automation
Tarski	EPL	Standalone	Separate model	Separate	Framework (Eclipse)	TIM	Link generation
Eclipse Capra	EPL	Standalone	Separate model	Separate	Framework (Eclipse)	TIM, Adapters, Reporting	Consistency checks
RecCycle	EPL	Requirements Management	Separate model	Hybrid	Framework (Eclipse)	TIM	None
OpenTrace	AGPL	Link recovery	Inline	Separate	Tool-chain specific (GATE)	Reporting	Link generation
OpenCert	OSS	Special Purpose (Safety certification)	Centralised	Hybrid	Framework (Eclipse)	Reporting	Unknown

After setting up the tool, an initial workshop with the researchers, the BA and the lead developer was conducted. This revealed an important aspect: some projects only had requirements, tickets, source code and tests, but were missing design models. We had two options to solve this challenge: 1) to add support in the TIM to link tickets to code; or 2) to create the missing design models. The BA and lead developer decided to create the missing design models since one of the best practices for the company is to have such models for all projects in order to facilitate comprehension of the system without referring to code. Enforcing this best practice through the TIM ensures consistency and adds an additional incentive for the company to maintain the design models.

In order to reduce the effort of creating the models, we reverse-engineered the current source code and created a UML model and relevant diagrams using the existing PHP code and BOUML [264]. The resulting UML models were imported into Papyrus[8] and thus became viewable and editable within Eclipse.

Once the development artifacts were in place, the BA selected one project to work on. For this project baseline metrics were noted down so that they can be used for comparison later on. Using Eclipse Capra, the BA imported all the artifacts relevant to the project, i.e., requirements, design models, code, and tests and created four types of links: 1) from requirements defined in Google spreadsheets to tickets in JIRA; 2) from requirements to model elements in UML; 3) from model elements to implementation code written in PHP; and 4) from implementation to tests which were also written in PHP.

At the end of the first day, the two researchers and the BA had a meeting to discuss if the process and the resulting links are sufficient. As shown in Figure 7.5, all the links were from requirements to other artifacts, including tickets. However, the developers use tickets and not the requirements during the sprints. The developers therefore needed to know which artifacts are related to a single ticket and not necessarily to the whole requirement. We therefore modified the TIM and made it “ticket-centric”. This is depicted in Figure 7.6, where a requirement is linked to a ticket and the rest of the development artifacts are linked from a ticket. While the deployed TIM was already assessed using the scenarios in a “dry run" manner in step 5, the need for this change was only visible once the tool was deployed and actual trace links were created.

From day two to day five, the BA continued to create the links while the researchers were present to fix any issues that arise. The researchers also observed how the team worked and conducted interviews and focus group meetings with the team members as a first step towards evaluating the traceability process. Details on evaluation are given in Section 7.5.7.

7.5.7 Step 9: Evaluate process and tool

For an initial evaluation during the deployment stage, we used focus groups and interviews with the members of the development team to evaluate both the process and tool. The evaluation started on the second day of the deployment week. During the stand-up meeting, the business analyst showed examples of the links to the team and asked them for feedback. The researchers took note of the feedback from the team and met with the business analyst afterwards to discuss the needed changes. The developers explained that the links were too

[8] `https://www.eclipse.org/papyrus/`

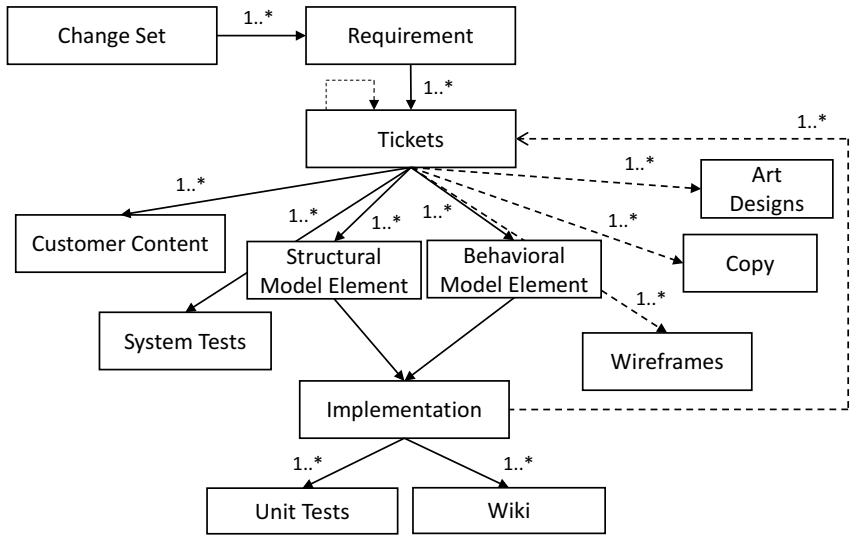


Figure 7.6: Evolved traceability information model. Links shown as dotted lines are captured in JIRA. Note that all development artifacts are now linked via the *Ticket* and the roadmap has been removed.

fine grained and that they preferred links on a higher level of granularity, for instance to link to the class and not to the method in the PHP code. Based on the evaluation of day one and two, we made changes to the TIM and the granularity of the links. To further evaluate the deployed process and tool, we also interviewed the product owner, in order to get his opinion on the process and the links that were created by the BA. We used an interview guide that we defined and made available online[9]. Additionally, we conducted a focus group meeting with the BA, two front-end developers and the lead developer to discuss and prioritise the previously elicited goals and how the new process would help achieve them. Based on the feedback collected from the interview with the product owner and focus group, we further customized the tool and provided a new version to the company. The changes made to the tool were mainly bug fixes. Early evaluation helped us tailor the process and tool, as exemplified by how the TIM evolved.

For the later evaluation steps, we used the metrics defined in Steps 1 and 2 of TracIMo. After two weeks, we conducted an interview with the BA via Skype to discuss how the links were used, which qualitative short-term benefits were evident, and if the company was facing any issues with the process or the tool. Five months after the pilot deployment, we conducted two additional interviews, one with the business analyst and one with a developer. These interviews investigated the benefits and challenges brought by the new traceability process. All the interviews were recorded, transcribed and analysed. Evaluation after several sprints enabled us to elicit short-term benefits of traceability. We also collected quantitative data from JIRA tickets. From January 2017 to May 2017, we collected data for 134 tickets, associated with four projects. After

[9] `https://tinyurl.com/y3n96ldq`

the introduction of traceability, we collected data for 17 tickets in which trace links were used associated with the same projects. Since the sample size of the latter tickets is small, we do not perform statistical analysis. However, we use the data to indicate trends. We identified the following benefits:

Estimation of tasks One of the challenges at the company was difficulty in estimating how much effort a task will need. In the interviews, the business analyst and one developer reported that the links embedded in the tickets made task estimation easier and more accurate since the developers could now not only see how many elements are associated with the tickets, but also *which* elements these are (e.g., classes, methods, tests, etc.).

Additionally, the developer reported improved estimation especially for tasks that affect third-party libraries, as such dependencies were not visible without the trace links. Table 7.8 shows that the number of incorrectly estimated tickets slightly decreased for three projects (A, B and C) after the introduction of traceability. With support from the qualitative data from the interviews, this is an indication that Traceability Goal 3 is met by the current traceability process. However, we will need to collect more data to be able to quantify this improvement.

Task Understandability The developer reported that the traceability graph embedded in the ticket makes tasks more understandable. The traceability graph is beneficial for novice developers, as they can see which artifacts are affected by the task and how these artifacts are connected:

"The advantage [of the new traceability approach] is, you can see which part of the system or the communication between the models and some parts of your code [are related to the task]. So it is some kind of visualization and makes it easy to understand." – [Developer]

Trace links to the requirements help developers understand the rationale of the different tasks. Our metrics from JIRA show the number of comments decreased after the introduction of traceability (cf. Table 7.8). A further analysis of these comments showed a decrease in the number of comments that suggested changes to the tickets or discussed dependency issues, indicating that developers understand the tasks and do not have to discuss them further. This is in line with Traceability Goals 2, 4 and 5. However, as previously mentioned, more data is needed to confirm this.

Detecting missing artifacts Through the links, the development team was able to identify missing artifacts. This was reported by the BA after a sprint planning meeting. If a ticket is, e.g., linked to a model element and this model element to implementation but not to tests, the latter are missing. The developers still have to investigate whether the tests were required or not but this investigation is simplified since the relevant elements are already identified.

7.5.8 Step 10: Anchor process and tool

The traceability strategy and tool needs to be anchored at the company. After the pilot study, the company needed to define how this new strategy will be adopted by all the development projects at the company. While this anchoring

Table 7.8: Selected metrics from the JIRA Ticketing system before and after the introduction of traceability.

Project	Total no. of tickets before	Total no. of tickets after	Wrong estimates before	Wrong estimates after	No. of comments before	No. of comments after
A	10	2	2	0	6	0
B	5	5	2	1	13	1
C	117	9	25	1	124	6
D	2	1	0	1	1	0

step is very important to ensure long-term benefits of traceability in the company and to develop the capabilities within the organisation, we could unfortunately not follow this process to its conclusion since the company was acquired and there was a change of personnel which hindered the progress of the project.

7.6 Discussion

In this section, we discuss the results of the study with respect to the research question stated in Section 7.1:

RQ: How can traceability be established to achieve the goals of the organisation and yield a measurable impact?

There are two parts to the research question: the first part is about how to design a traceability strategy and the second part is about ensuring that the designed strategy yields measurable benefits. We designed TracIMo which guarantees both. In this section we discuss key points w.r.t. to designing a traceability strategy in Section 7.6.1 and measuring the impact in Section 7.6.2. Additionally, we encountered several challenges as a result of the traceability strategy we designed. These challenges are discussed in Section 7.6.3.

7.6.1 Designing a tailored traceability strategy

The study proposes a methodology to define a traceability strategy for software development organisations. The steps in this methodology (cf. Figure 7.2) are geared towards analyzing the needs of the company and making sure that the specific traceability strategy is tailored accordingly, regardless of the development process used. Steps 1 to 5 of the framework have already been shown to be effective in practice [246] for assessing traceability strategies. We extended these steps and added steps 6 to 10 to allow us to define and refine a traceability strategy for a development team that is used in practice.

A particular strength of the proposed method is the alignment between the process goals and the traceability goals. By using GQM [84], an established technique from software process improvement, we were able to achieve both aspects. As shown in Tables 7.5.2 and 7.4, this thorough analysis allowed us to define a traceability information model that is specific for the company.

It also ensured the traceability goals and process goals are compatible. In communicating with the company, the clearly defined goals also allowed us to discuss the scope of the changes and what is realistically achievable with traceability and gave us a way to evaluate the defined traceability strategy.

We also exploited GQM's strengths by defining metrics that allowed us to measure the benefits of traceability. Each goal was associated with a number of metrics. For some of them, data was collected as a baseline before the introduction of traceability. Compared with data after traceability was introduced, advantages could be identified as shown in Table 7.8.

The iterations built into phase 2 of the method also proved helpful. This is particularly evident in the evolution of the traceability information model. We discovered the issues with the TIM only through deployment and evaluation of traceability in practice. Since the method was iterative and a second iteration was planned for the time the researchers were present at the company, the issues could be quickly addressed.

Using TracIMo, we were able to design a tailored traceability strategy that fits the company's agile team needs. Several studies discuss traceability for plan-driven development processes (e.g. [7, 17, 58]), where traceability is focused on development artifacts that are assumed to be persisted and maintained over the development life-cycle. Traceability is therefore a requirements-centered activity where links are created from requirements to other development artifacts like design models and code [14,108]. In this agile context, however, the development is driven by the tickets rather than the requirements. Even though tickets are derived from requirements, developers are used to dealing with tickets. The lifetime of a ticket is the sprint(s) where it is worked on. When a ticket is marked as done, developers do not look at it again. We believe this is the case for many agile projects [265]. We tailored the traceability process to the development process of the company by defining a ticket-centric traceability strategy. As shown in Figure 7.6, tickets are linked to development artifacts such as requirements, design models and transitively to code and tests. This means that the links created are specific to a specific ticket. The advantage of this task-centric traceability approach is that it allows for incremental creation of links in situations where links are created retrospectively. However, as the trace model grows, filtering mechanisms are needed since existing links between development artifacts which were created with previous tickets may not be relevant for current tickets. In our case, we implemented a filtering mechanism that allowed the BA to filter out unnecessary links before attaching the trace links graph in the tickets.

7.6.2 Measuring the impact of the traceability strategy

From the interviews and our measurements, we gathered qualitative and quantitative data to support three benefits: 1) improvement in effort estimation, 2) improvement in task understandability and 3) improvement in identification of missing artifacts, as reported in Section 7.5.

As discussed in [30], defining traceability goals and ensuring that a company captures the information required to fulfill these goals is a first step towards ensuring the return on investment (RoI) of traceability. We observed in our case that the benefits we elicited are due to Goal 2 (improve visibility of

decision rationale), Goal 3 (improve accuracy of effort estimation), Goals 4 and 5 (increase efficiency of identifying artifacts relevant to a change). While the designed TIM and process are aimed to fulfill all goals, further evaluation is needed to elicit the benefits of Goals 1, 6 and 7.

Having said this, one of the major challenges of traceability is the inability to measure its RoI [8, 19]. This is because the benefits of traceability require time to manifest and may be affected by other factors such as the type of project and employee turnover [30]. It is also difficult to determine the entire cost of traceability in the development life-cycle [19]. One of our long-term goals was to investigate the RoI of traceability for the company, however, due to organisational changes that occurred at the company, this data collection is no longer possible. While it is possible to quantify the amount of effort invested to design the traceability strategy, deploy the strategy at the company and the average amount of time it takes the BA to create links, even with these kinds of measurements, it is difficult to quantify the RoI after a short time. It was only possible to observe perceived benefits in a qualitative manner using the follow-up interviews.

To make sure that the amount of time invested in applying TracIMo is manageable, we provide our recommendations on how to effectively apply TracIMo in Section 7.6.4.

7.6.3 Challenges of traceability

We encountered five challenges as well as important decisions that needed to be made during the design of the traceability strategy:

[a] trace link granularity;

[b] scope of the trace links;

[c] the need for intermediate artifacts;

[d] time required to create links; and

[e] adoption of the traceability process.

While these challenges are not new to the research community, we discuss how we encountered them and dealt with them in order to provide more practical insights for both practitioners and researchers.

Trace link granularity Several studies (e.g., [82, 118, 123]) report that it is difficult for companies to know the right level of granularity for the trace links. In our study we also encountered this challenge. This was especially tricky for design artifacts (models) and implementation artifacts (code). During the first day of deployment, links were created as fine-grained as possible. A ticket was, e.g., linked to a specific UML attribute in a UML class and to a specific PHP method in a PHP class. The feedback from developers was that there were too many links, making the traceability graph difficult to understand. The development team suggested to use more coarse-grained links on the class level for both the models and the code. However, in the follow-up interview, the BA reported that there are still tickets for which it makes sense

to create links to detailed design and implementation. We thus decided that the granularity of the links will be determined by the granularity of the ticket. If the ticket contains low-level implementation details, then it will be linked to detailed implementation and design and vice versa. As a rule of thumb, the granularity between the connected artifacts should match [123]. As a consequence, traceability tools and the TIM should provide support for linking to different levels of granularity so that users are flexible.

Scope of trace links While links are created with respect to specific tickets, the traceability graph for a certain model element shows all existing links. If ticket A is, e.g., linked to model element B, but model element B was previously linked to ticket C, the developers see all this information in the traceability graph. This can be confusing as the developer is only interested in links to the ticket she is working on. To overcome this challenge, we developed filtering mechanisms that limit the links to those related to the ticket. This was done by making sure that the traceability graph contains links only to a selected ticket. While this solution worked for the company, more sophisticated solutions exist. For instance, the traceability tool Yakindu Traceability [266] provides a query language that can be used to query the trace model depending on what links the user is interested in. Additionally research to process unstructured natural language trace queries [267] and visual trace queries [268] also exist.

Introduction of intermediate development artifacts for traceability purposes As described in Section 7.5, supporting the traceability goals and using the TIM as intended required to introduce UML models of the current software in some projects. As a consequence, the BA now needs to introduce new model elements that are necessary to fulfill a requirement. This is necessary to show the new elements in the traceability graphs. The company will thus make the models the gold-standard and introduce new elements in the model before they are implemented. A potential drawback of this approach is that model and source code might get out of sync and therefore the model will not be used. To solve this, notification mechanisms need to be put in place to notify the BA of new classes that do not exist in the model. Such mechanisms could automatically detect changes in the source code and send a summary of these to the BA to incorporate corresponding changes in the UML model.

In more general terms, achieving traceability goals might make it necessary to create new types of artifacts that need to be maintained and integrated into the process. This can be costly and might require additional changes to roles, activities, and processes. In the case of the organisation, using a UML model of the entire software was considered best practice, so that the creation of the full UML model was considered a positive side aspect. In other cases, however, the introduction of new development artifacts can be a liability and the overall cost of introducing models into a development process is very hard to estimate [269].

Time taken to create links Creating trace links in retrospect when plenty of development artifacts already exist is a time consuming task. For instance, we measured that it took the BA approximately 30 minutes to create seven links to one ticket, which means an average of 4.2 minutes to create one link.

Note that this time involves the time to decide on what needs to be linked and to locate the artifacts to be linked. This is a well known traceability problem [14]. There is research on automation of this process (e.g., [21,25,226]) but the resulting links are not 100% correct and have to be checked manually, which is also a time consuming task especially if the tool produces many false positives [220]. Since trace links are created for specific tickets in this case, the BA does not need to create all links at once. It is sufficient if the developers have links for the tasks they are working on in a particular sprint. This means that the task of creating links can be performed incrementally and is therefore manageable for the BA.

Adoption of the traceability process We faced some resistance by the lead developer who did not make time for creating or using trace links. This is because the lead developer had a lot of experience in the system. Even though in the interview he showed an interest in traceability, he did not have an immediate need for trace links and therefore was not motivated to create them. He was not the main beneficiary of the trace links, either, but still one of the best candidates to create the links due to his experience in the system. Resistance to change is a well known challenge in change management literature [270]. Specifically for traceability, the creators of the links are usually not the ones who benefit the most since they already know the system well [82]. This serves as a reminder that for each change introduced in a company, it is crucial to make sure that all people who will be affected are involved in the change. It is also important to ensure that all the involved stakeholders understand clearly what the change is and how they will benefit from it.

7.6.4 Reflections on applying TracIMo

In this section, we give our reflections on the experience of applying TracIMo to the company. Since TracIMo consists of ten steps and may seem like a heavy weight approach, we give the following four recommendations on how it can be applied effectively.

Carry out several steps at the same time. While TracIMo consists of ten distinct steps, in a realistic setting, some of these steps can be carried out together in order to leverage the synergies between them. For instance, Steps 1 and 2 both use data from the development process and can be carried out together. The same is true for Steps 3 and 5.

Choose the right roles Applying TracIMo in a company requires data from different roles. It is important to choose these roles with care in the beginning in order to reduce the number of iterations needed to design a working traceability strategy. For instance, in our case, we had the BA as the main point of contact. However, we also interviewed developers and product owners in order to get the full picture at the company. In cases where TracIMo will be used without the help of researchers (which is what we envision), an experienced person with a senior/managerial role at the company with intimate knowledge of the development process as well as the developed product should take the lead in conducting the steps. This has the advantage that the person already has a lot

of information required by TracIMo and will thus reduce the time needed to perform some of the steps that require data collection. Care has to be taken, though, that all stakeholders are included and implicit biases do not yield an unsuitable traceability strategy.

Define metrics based on available data TracIMo requires the definition of metrics in order to measure how the defined traceability strategy is performing. It may be tempting to define metrics whose data is not yet available and for which systematic measurements need to be established. While these metrics may prove useful, this is recommended if and only if there is no alternative data available that can be used to measure that particular aspect. We recommend to define metrics that use already available data in the development process, or data that can be automatically collected to reduce the amount of effort needed in data collection.

Mind the level of formality For steps that require documentation, TracIMo gives recommendations on which notations are available. For instance when defining the traceability process, on one end of the spectrum, it is possible to use a formal language such as SPEM and on the other end, one can use wikis to document the process. In a realistic setting, we recommend that the level of formality matches with what is expected in the organisation. For example if an organisation follows agile principles where there is a need for little documentation, the traceability strategy can be lightly documented. However, if a company is in a safety critical domain and requires the process to be formally documented, a formal language can be used. This is to ensure that the amount of effort spent on defining the traceability strategy is minimized.

7.7 Threats to Validity

In this section we describe the limitations of our study first with respect to the design of TracIMo and second with respect to how TracIMo was evaluated.

To design TracIMo, we modified and extended the steps in Rempel et al.'s [246] methodology and added our own. When reasoning about which steps are needed, our aim was to make sure that we cover all the steps needed to design, implement and evaluate a traceability strategy. To verify that the methodology makes sense we used a number of brainstorming sessions with the researchers and the BA from the company. As such, there is a chance that the methodology may be lacking some steps that are specific to other contexts. The company we conducted the study with is small, has one small development team and uses agile development methodologies in their development process. Therefore, TracIMo needs to be applied in other contexts to verify both it's applicability and generalizability.

With respect to evaluation of TracIMo, we used a case study in our design science cycle where we designed a traceability strategy using TracIMo at a company. There a number of threats to validity applicable to the case study which are discussed below using the categories defined by Runeson and Höst [64].

Construct validity Construct validity aims to verify that the concepts that are researched are understood by subjects of the research. To evaluate the designed traceability strategy, we had multiple interviews and focus groups after the introduction of traceability. To make sure that the interviewees understood the concepts we were researching we introduced the topic of traceability to all respondents before the interview and gave examples. We also performed member checking with the BA to verify the data from the interviews.

Internal validity Internal validity is relevant when a causal relationship is investigated. The immediate benefits of traceability constitute such a causal relationship. Researchers have to make sure that there are no other factors that could affect this investigated relationship. While there were several speculated benefits, we only reported benefits where other influencing factors could be excluded. Additionally, during the study, the company went through several changes: 1) a change in the development process (from an isolated development team to a cross cutting development team); 2) a merger with another company; and 3) one of the developers left the company. While we continued the study according to the planned methodology, the changes in the company may have an effect on our results, especially since developers had less time to work with the links during the merger. This also led to less data being available for quantitative evaluation. Additionally, the lead developer was reluctant to create trace links. We attribute this to the fact that he already knew the system well and thought that trace links would not be useful for him but only for the other developers.

External validity External validity refers to how the results of the study can be generalized. Since we evaluated the methodology with one case study in one company, the particularities of this company might have been conducive to the application of TracIMo. We believe that the steps in TracIMo are generic enough and independent of the context, however, further case studies are needed to verify this. The results, however, including the process and traceability goals, the TIM supporting these goals, and the concrete steps in the process are specific to the case company. Since design science is an iterative process, we believe additional validation in other organisations is thus needed to conclusively proof TracIMo's generalizability.

Reliability This validity threats refers to whether the study is repeatable. We have documented our case study process as much as possible. For instance, our interview guide[10] and the detailed description of the case study[11] are also available online. This is to ensure that other researchers who want to repeat the study have all the materials they need and to allow practitioners to use TracIMo as a basis for defining a tailored traceability strategy for their organisation.

[10] `https://tinyurl.com/y3n96ldq`
[11] `https://tinyurl.com/y6dmd8u9`

7.8 Conclusions and Future Work

This paper presents TracIMo, a methodology to systematically design and introduce a traceability strategy in companies. It describes the different steps in the methodology and demonstrates how this can be applied in practice through an industrial case study. While TracIMo can be used to introduce traceability in development companies employing different development processes, we conducted our study with an agile development team. This led to the creation of a "ticket-centric" and incremental traceability strategy can be used to effectively create trace links in retrospect.

Our study also revealed a number of benefits of traceability which can serve as an encouragement to companies thinking about adopting traceability. The main takeaway is that, in order to gain benefits from traceability, it is crucial to define specific traceability goals upfront, and design a traceability strategy that will enable the development team to reach these goals. This requires tailoring the traceability information model and the traceability tool and deriving metrics that to measure how the goals have been fulfilled.

While we encountered several challenges and have proposed solutions for these challenges, further validation of these solutions is needed. As part of our future work, we plan to further evaluate the benefits of traceability in order to elicit long term benefits of our approach and devise strategies to quantitatively measure the return on investment of traceability, from the set of metrics derived. Additionally, since TracIMo was only evaluated in one case, we plan to replicate the study with other companies to further check the feasibility of the methodology.

Bibliography

[1] N. Heumesser and F. Houdek, "Experiences in managing an automotive requirements engineering process," in *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International.* IEEE, 2004, pp. 322–327.

[2] R. Purushothaman and D. E. Perry, "Toward understanding the rhetoric of small source code changes," *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 511–526, 2005.

[3] B. Regnell, R. B. Svensson, and K. Wnuk, "Can we beat the complexity of very large-scale requirements engineering?" in *International Working Conference on Requirements Engineering: Foundation for Software Quality.* Springer, 2008, pp. 123–128.

[4] E. Bouillon, P. Mäder, and I. Philippow, "A survey on usage scenarios for requirements traceability in practice," in *International Working Conference on Requirements Engineering: Foundation for Software Quality.* Springer, 2013, pp. 158–173.

[5] G. Spanoudakis and A. Zisman, "Software traceability: a roadmap," *Handbook of Software Engineering and Knowledge Engineering*, vol. 3, pp. 395–428, 2005.

[6] S. Ibrahim, N. B. Idris, M. Munro, and A. Deraman, "A software traceability validation for change impact analysis of object oriented software." in *Software Engineering Research and Practice*, 2006, pp. 453–459.

[7] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE transactions on software engineering*, vol. 27, no. 1, pp. 58–93, 2001.

[8] G. Regan, F. McCaffery, K. McDaid, and D. Flood, "The barriers to traceability and their potential solutions: Towards a reference framework," in *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on.* IEEE, 2012, pp. 319–322.

[9] R. Torkar, T. Gorschek, R. Feldt, M. Svahnberg, U. A. Raja, and K. Kamran, "Requirements traceability: a systematic review and industry case study," *International Journal of Software Engineering and Knowledge Engineering*, vol. 22, no. 03, pp. 385–433, 2012.

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

[11] COEST, "Center of excellence for software traceability (coest)," 2015, accessed : 2016-05-18. [Online]. Available: http://www.coest.org

[12] "Ieee recommended practice for software requirements specifications," *IEEE Std 830-1998*, pp. 1–40, Oct 1998.

[13] J. Radatz, A. Geraci, and F. Katki, "Ieee standard glossary of software engineering terminology," *IEEE Std*, vol. 610121990, no. 121990, p. 3, 1990.

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

[15] G. Spanoudakis, "Plausible and adaptive requirement traceability structures," in *Proceedings of the 14th international conference on Software engineering and knowledge engineering*. ACM, 2002, pp. 135–142.

[16] O. Gotel and P. Mäder, "Acquiring tool support for traceability," in *Software and systems traceability*. Springer, 2012, pp. 43–68.

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

[18] R. Dömges and K. Pohl, "Adapting traceability environments to project-specific needs," *Commun. ACM*, vol. 41, no. 12, pp. 54–62, Dec. 1998. [Online]. Available: http://doi.acm.org.proxy.lib.chalmers.se/10.1145/290133.290149

[19] O. Gotel, J. Cleland-Huang, J. H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, and J. Maletic, "The grand challenge of traceability (v1. 0)," in *Software and Systems Traceability*. Springer, 2012, pp. 343–409.

[20] A. D. Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Recovering traceability links in software artifact management systems using information retrieval methods," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 16, no. 4, p. 13, 2007.

[21] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, "A machine learning approach for tracing regulatory codes to product specific requirements," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. ACM, 2010, pp. 155–164.

[22] 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.

[23] G. Spanoudakis, A. Zisman, E. Pérez-Minana, and P. Krause, "Rule-based generation of requirements traceability relations," *Journal of Systems and Software*, vol. 72, no. 2, pp. 105–127, 2004.

[24] J. Guo, J. Cheng, and J. Cleland-Huang, "Semantically enhanced software traceability using deep learning techniques," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, pp. 3–14.

[25] M. Borg, P. Runeson, and A. Ardö, "Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability," *Empirical Software Engineering*, vol. 19, no. 6, pp. 1565–1616, 2014.

[26] S. Maro, A. Anjorin, R. Wohlrab, and J.-P. Steghöfer, "Traceability maintenance: factors and guidelines," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2016, pp. 414–425.

[27] J. Holtmann, J.-P. Steghöfer, M. Rath, and D. Schmelter, "Cutting through the Jungle: Disambiguating Model-based Traceability Terminology," in *Proceedings of the 28th IEEE International Requirements Engineering Conference (REâĂŹ20)*. IEEE, 2020, p. 10.

[28] J. Cleland-Huang, B. Berenbach, S. Clark, R. Settimi, and E. Romanova, "Best practices for automated traceability," *Computer*, vol. 40, no. 6, 2007.

[29] P. Mäder, O. Gotel, and I. Philippow, "Enabling automated traceability maintenance through the upkeep of traceability relations," in *Model Driven Architecture-Foundations and Applications*. Springer, 2009, pp. 174–189.

[30] C. Ingram and S. Riddle, "Cost-benefits of traceability," in *Software and Systems Traceability*. Springer, 2012, pp. 23–42.

[31] P. Rempel, P. Mäder, T. Kuschke, and I. Philippow, "Requirements traceability across organizational boundaries-a survey and taxonomy." in *REFSQ*. Springer, 2013, pp. 125–140.

[32] V. Kirova, N. Kirby, D. Kothari, and G. Childress, "Effective requirements traceability: Models, tools, and practices," *Bell Labs Technical Journal*, vol. 12, no. 4, pp. 143–157, 2008.

[33] open services.net, "Open services for lifecycle collaboration," 2017, accessed : 2017-03-20. [Online]. Available: https://open-services.net

[34] J. Cleland-Huang, O. C. Gotel, J. Huffman Hayes, P. Mäder, and A. Zisman, "Software traceability: trends and future directions," in *Proceedings of the on Future of Software Engineering*. ACM, 2014, pp. 55–69.

[35] N. Skrypuch, "Eclipse modeling-emf-home," 2014, accessed : 2014-06-13. [Online]. Available: http://www.eclipse.org/modeling/emf/

[36] D. Cuddeback, A. Dekhtyar, and J. H. Hayes, "Automated requirements traceability: The study of human analysts," in *Requirements Engineering Conference (RE), 2010 18th IEEE International*. IEEE, 2010, pp. 231–240.

[37] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining" gamification"," in *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, 2011, pp. 9–15.

[38] A. Kannenberg and H. Saiedian, "Why software requirements traceability remains a challenge," *CrossTalk The Journal of Defense Software Engineering*, vol. 22, no. 5, pp. 14–19, 2009.

[39] A. Marcus and J. I. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," in *25th International Conference on Software Engineering, 2003. Proceedings.* IEEE, 2003, pp. 125–135.

[40] P. Hübner and B. Paech, "Increasing precision of automatically generated trace links," in *International Working Conference on Requirements Engineering: Foundation for Software Quality.* Springer, 2019, pp. 73–89.

[41] P. Hübner and B. Paech, "Using interaction data for continuous creation of trace links between source code and requirements in issue tracking systems," in *International Working Conference on Requirements Engineering: Foundation for Software Quality.* Springer, 2017, pp. 291–307.

[42] A. Dekhtyar, O. Dekhtyar, J. Holden, J. H. Hayes, D. Cuddeback, and W.-K. Kong, "On human analyst performance in assisted requirements tracing: Statistical analysis," in *2011 IEEE 19th International Requirements Engineering Conference.* IEEE, 2011, pp. 111–120.

[43] D. Cuddeback, A. Dekhtyar, J. H. Hayes, J. Holden, and W. K. Kong, "Towards overcoming human analyst fallibility in the requirements tracing process: Nier track," in *33rd Int. Conf. on Software Engineering (ICSE '11)*, May 2011, pp. 860–863.

[44] W.-K. Kong, J. Huffman Hayes, A. Dekhtyar, and J. Holden, "How do we trace requirements: an initial study of analyst behavior in trace validation tasks," in *4th Int. Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '11).* ACM, 2011, pp. 32–39.

[45] J. H. Hayes, A. Dekhtyar, S. K. Sundaram, E. A. Holbrook, S. Vadlamudi, and A. April, "Requirements tracing on target (retro): improving software maintenance through traceability recovery," *Innovations in Systems and Software Engineering*, vol. 3, no. 3, pp. 193–202, 2007.

[46] J. Lin, C. C. Lin, J. Cleland-Huang, R. Settimi, J. Amaya, G. Bedford, B. Berenbach, O. B. Khadra, C. Duan, and X. Zou, "Poirot: A distributed tool supporting enterprise-wide automated traceability," in *14th IEEE Int. Requirements Engineering Conf. (RE' 06).* IEEE, 2006, pp. 363–364.

[47] G. Bavota, L. Colangelo, A. De Lucia, S. Fusco, R. Oliveto, and A. Panichella, "Traceme: traceability management in eclipse," in *28th IEEE Int. Conf. on Software Maintenance (ICSM '12)*. IEEE, 2012, pp. 642–645.

[48] P. Rempel, S. Lehnert, T. Kuschke *et al.*, "A framework for traceability tool comparison," *Softwaretechnik-Trends*, vol. 32, no. 3, pp. 6–11, 2012.

[49] P. Mader, O. Gotel, and I. Philippow, "Enabling automated traceability maintenance by recognizing development activities applied to models," in *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2008, pp. 49–58.

[50] P. Mäder, O. Gotel, and I. Philippow, "Rule-based maintenance of post-requirements traceability relations," in *2008 16th IEEE International Requirements Engineering Conference*. IEEE, 2008, pp. 23–32.

[51] M. Rahimi and J. Cleland-Huang, "Evolving software trace links between requirements and source code," *Empirical Software Engineering*, vol. 23, no. 4, pp. 2198–2231, 2018.

[52] F. Erata, M. Challenger, B. Tekinerdogan, A. Monceaux, E. Tüzün, and G. Kardas, "Tarski: A platform for automated analysis of dynamically configurable traceability semantics," in *Proceedings of the Symposium on Applied Computing*, 2017, pp. 1607–1614.

[53] S. Bode, S. Lehnert, and M. Riebisch, "Comprehensive model integration for dependency identification with emftrace," in *Joint Proc. of the First Int. Workshop on Model-Driven Software Migration (MDSM 2011) and the Fifth Int. Workshop on Software Quality and Maintainability (SQM 2011)*, 2011, pp. 17–20.

[54] Z. ITU-T, "151 user requirements notation (urn)–language definition," *ITU-T, Nov*, 2008.

[55] P. Rempel and P. Mäder, "A quality model for the systematic assessment of requirements traceability," in *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. IEEE, 2015, pp. 176–185.

[56] A. Espinoza and J. Garbajosa, "A study to support agile methods more effectively through traceability," *Innovations in Systems and Software Engineering*, vol. 7, no. 1, pp. 53–69, 2011.

[57] U. Durrani, J. Richardson, J. Lenarcic, and Z. Pita, "Lean traceability solution through slam model," in *22nd Australasian Software Engineering Conference-Industry Proceedings, Melbourne*, 2013.

[58] H. U. Asuncion, F. François, and R. N. Taylor, "An end-to-end industrial software traceability tool," in *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. ACM, 2007, pp. 115–124.

[59] I. Galvao and A. Goknil, "Survey of traceability approaches in model-driven engineering," in *Enterprise Distributed Object Computing Conference (EDOC'07)*. IEEE, 2007, pp. 313–324.

[60] I. Santiago, A. Jiménez, J. M. Vara, V. De Castro, V. A. Bollati, and E. Marcos, "Model-driven engineering as a new landscape for traceability management: A systematic literature review," *Information and Software Technology*, vol. 54, no. 12, pp. 1340–1356, 2012.

[61] S. Nair, J. L. de la Vara, and S. Sen, "A review of traceability research at the requirements engineering conference re@ 21," in *Requirements Engineering Conference (RE), 2013 21st IEEE International*. IEEE, 2013, pp. 222–229.

[62] S. Keele, "Guidelines for performing systematic literature reviews in software engineering," in *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*. sn, 2007.

[63] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Information and Software Technology*, vol. 106, pp. 101–121, 2019.

[64] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[65] VDA QMC Working Group 13 / Automotive SIG, "Automotive SPICE Process Assessment / Reference Model," Automotive Special Interest Group, Tech. Rep., 2015.

[66] K. Lukka, *The Constructive Research Approach*, 01 2003, pp. 83–101.

[67] G. D. Crnkovic, "Constructive research and info-computational knowledge generation," in *Model-Based Reasoning in Science and Technology*. Springer, 2010, pp. 359–380.

[68] R. Wieringa, "Design science methodology: principles and practice," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*. ACM, 2010, pp. 493–494.

[69] S. Maro and J.-P. Steghöfer, "Capra: A configurable and extendable traceability management tool," in *Requirements Engineering Conference (RE), 2016 IEEE 24th International*. IEEE, 2016, pp. 407–408.

[70] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 285–311.

[71] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[72] R. K. Yin, *Case Study Research: Design and Methods. 3rd edition*. Thousand Oaks, CA: Sage, 2003.

[73] D. Falessi, N. Juristo, C. Wohlin, B. Turhan, J. Münch, A. Jedlitschka, and M. Oivo, "Empirical software engineering experts on the use of students and professionals in experiments," *Empirical Software Engineering*, vol. 23, no. 1, pp. 452–489, 2018.

[74] J. W. Creswell and D. L. Miller, "Determining validity in qualitative inquiry," *Theory into practice*, vol. 39, no. 3, pp. 124–130, 2000.

[75] International Organization for Standardization, "Road vehicles – functional safety," *ISO26262:2011*, Nov. 2011.

[76] Plantuml.sourceforge.net, "Plantuml," 2015, accessed : 2015-06-01. [Online]. Available: http://plantuml.sourceforge.net

[77] Eclipse.org, "Graphical editing framework," 2017, accessed: 2017-03-13. [Online]. Available: https://eclipse.org/gef/

[78] J. H. Hayes, A. Dekhtyar, J. Larsen, and Y.-G. Gueheneuc, "Effective use of analysts effort in automated tracing," *Requirements Engineering*, vol. 23, no. 1, pp. 119–143, 2018.

[79] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *International symposium on handheld and ubiquitous computing*. Springer, 1999, pp. 304–307.

[80] R. M. Parizi, S. P. Lee, and M. Dabbagh, "Achievements and challenges in state-of-the-art software traceability between test and code artifacts," *IEEE Transactions on Reliability*, vol. 63, no. 4, pp. 913–926, 2014.

[81] O. Pedreira, F. García, N. Brisaboa, and M. Piattini, "Gamification in software engineering–a systematic mapping," *Information and Software Technology*, vol. 57, pp. 157–168, 2015.

[82] S. Maro, J.-P. Steghöfer, and M. Staron, "Software traceability in the automotive domain: Challenges and solutions," *Journal of Systems and Software*, vol. 141, pp. 85–110, 2018.

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

[84] R. Van Solingen, V. Basili, G. Caldiera, and H. D. Rombach, "Goal/Question/Metric (GQM) approach," *Encyclopedia of Software Engineering*, 2002.

[85] O. Gotel, J. Cleland-Huang, J. H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, and G. Antoniol, "The quest for ubiquity: A roadmap for software and systems traceability research," in *Requirements Engineering Conference (RE), 2012 20th IEEE International*. IEEE, 2012, pp. 71–80.

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

[87] M. Kassab, O. Ormandjieva, and M. Daneva, "A traceability metamodel for change management of non-functional requirements," in *2008 Sixth International Conference on Software Engineering Research, Management and Applications*. IEEE, 2008, pp. 245–254.

[88] J. Cleland-Huang, M. Heimdahl, J. H. Hayes, R. Lutz, and P. Maeder, "Trace queries for safety requirements in high assurance systems," in *International working conference on requirements engineering: Foundation for software quality*. Springer, 2012, pp. 179–193.

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

[90] M. Broy, "Challenges in automotive software engineering," in *Proceedings of the 28th international conference on Software engineering*. ACM, 2006, pp. 33–42.

[91] J. Dannenberg and J. Burgard, "Car innovation: A comprehensive study on innovation in the automotive industry," 2015.

[92] A. Busnelli, "Car Software: 100M Lines of Code and Counting," https://www.linkedin.com/pulse/20140626152045-3625632-car-software-100m-lines-of-code-and-counting, 2014, [Online; accessed 07-10-2016].

[93] 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)*, 2003.

[94] M. Born, J. Favaro, and O. Kath, "Application of ISO DIS 26262 in Practice," in *Proceedings of the 1st Workshop on Critical Automotive Applications: Robustness & Safety*, ser. CARS '10. New York, NY, USA: ACM, 2010, pp. 3–6.

[95] D. Sexton, A. Priore, and J. Botham, "Effective Functional Safety Concept Generation in the Context of ISO 26262," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, vol. 7, no. 1, pp. 95–102, 2014.

[96] F. Blaauboer, K. Sikkel, and M. N. Aydin, *Deciding to Adopt Requirements Traceability in Practice*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 294–308.

[97] S. Maro, M. Staron, and J.-P. Steghöfer, "Challenges of establishing traceability in the automotive domain," in *Proceedings of the 9th International Conference on Software Quality*. Springer, 2017.

[98] P. Oliveira, A. L. Ferreira, D. Dias, T. Pereira, P. Monteiro, and R. J. Machado, "An analysis of the commonality and differences between aspice and iso26262 in the context of software development," in *European*

Conference on Software Process Improvement. Springer, 2017, pp. 216–227.

[99] P. Johannessen, Ö. Halonen, and O. Örsmark, "Functional safety extensions to automotive spice according to iso 26262," in *International Conference on Software Process Improvement and Capability Determination*. Springer, 2011, pp. 52–63.

[100] A. Qusef, G. Bavota, R. Oliveto, A. D. Lucia, and D. Binkley, "Recovering test-to-code traceability using slicing and textual analysis," *Journal of Systems and Software*, vol. 88, pp. 147 – 168, 2014.

[101] M. A. Javed and U. Zdun, "The supportive effect of traceability links in change impact analysis for evolving architectures – two controlled experiments," in *Software Reuse for Dynamic Systems in the Cloud and Beyond: 14th International Conference on Software Reuse, ICSR 2015, Miami, FL, USA, January 4-6, 2015. Proceedings*, I. Schaefer and I. Stamelos, Eds. Cham: Springer International Publishing, 2014, pp. 139–155.

[102] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th international conference on evaluation and assessment in software engineering*, vol. 17, no. 1. sn, 2008, pp. 1–10.

[103] V. Garousi, M. Felderer, and M. V. Mäntylä, "The Need for Multivocal Literature Reviews in Software Engineering: Complementing Systematic Literature Reviews with Grey Literature," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '16. New York, NY, USA: ACM, 2016, pp. 26:1–26:6.

[104] S. F. Königs, G. Beier, A. Figge, and R. Stark, "Traceability in systems engineering–review of industrial practices, state-of-the-art technologies and new research solutions," *Advanced Engineering Informatics*, vol. 26, no. 4, pp. 924–940, 2012.

[105] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt," *Journal of Systems and Software*, vol. 86, no. 6, pp. 1498 – 1516, 2013.

[106] M. F. Bashir and M. A. Qadir, "Traceability techniques: A critical study," in *2006 IEEE International Multitopic Conference*. IEEE, 2006, pp. 265–268.

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

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

[109] A. Espinoza, P. P. Alarcon, and J. Garbajosa, "Analyzing and systematizing current traceability schemas," in *Software Engineering Workshop, 2006. SEW'06. 30th Annual IEEE/NASA*. IEEE, 2006, pp. 21–32.

[110] R. Oliveto, G. Antoniol, A. Marcus, and J. Hayes, "Software artefact traceability: the never-ending challenge," in *IEEE International Conference on Software Maintenance*. IEEE, 2007, pp. 485–488.

[111] D. D. Ward, "MIRA Safety Analysis - Vector," https://vector.com/portal/medien/cmc/events/commercial_events/VU_Conference/VUC13/VeCoUK13/VeCoUK13-Day2/Vector_FS_Conf_June13_DDW.pdf, 2013.

[112] J. Leuser, "Challenges for semi-automatic trace recovery in the automotive domain," in *2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, 2009.

[113] "Musterbasierte Analyse und Korrektur von Trace ... - UnivIS - FAU," https://univis.fau.de/formbot/dsc_3Danew_2Fresrep_view_26rprojs_3Dtech_2FIMMD_2Flsinfs_1_2Fmuster_26dir_3Dtech_2FIMMD_2Flsinfs_1_26ref_3Dresrep, 2008.

[114] E. Brandenburg, "USABILITY-OPTIMIZED TRACEABILITY FOR DEVELOPMENT OF EMBEDDED SYSTEM ACCORDING TO ISO 26262," http://optrac.de/wp-content/uploads/2014/06/Usability-Optimized-Traceability-for-Development-of-Embedded\\-Systems-according-to-ISO-26262.pdf, 2014.

[115] D. Seidler and T. Southworth, "ASPICE Made Easy-Case Studies and Lessons Learned - IBM," https://www.ibm.com/developerworks/community/files/basic/anonymous/api/library/9e66f5de-701e-4994-9291-75646d558240/document/d88a5328-3d2a-4e82-9a3f-e4e5bf9f41be/media/Session22-A-SPICE%20compliance%20made%20easy%20Case%20studies%20and%20lessons%20learned.pdf, 2013.

[116] "Software and Systems Traceability for Safety ... - Schloss Dagstuhl," http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=15162.

[117] A. Willert, "Traceability - Willert Software Tools GmbH," https://www.willert.de/assets/Newsletter/ESER-35-Traceability-in-Theorie-und-Praxis.pdf.

[118] M. A. Javed and U. Zdun, "A systematic literature review of traceability approaches between software architecture and source code," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2014, p. 16.

[119] A. Von Knethen and B. Paech, "A survey on tracing approaches in practice and research," *Frauenhofer Institut Experimentelles Software Engineering, IESE-Report No*, vol. 95, 2002.

[120] M.-A. Peraldi-Frati and A. Albinet, "Requirement traceability in safety critical systems," in *1st Workshop on Critical Automotive applications: Robustness & Safety (CARS 2010)*, 2010.

[121] J. Gonzalez-Huerta, E. Insfran, S. Abrahao, and J. McGregor, "Non-functional requirements in model-driven software product line engineering," in *4th International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages, NFPinDSML 2012*, 2012.

[122] N. Pavkovic, N. Bojcetic, L. Franic, and D. Marjanovic, "CASE STUDIES TO EXPLORE INDEXING ISSUES IN PRODUCT ..." in *DS 68-6: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 6: Design Information and Knowledge, Lyngby/Copenhagen, Denmark, 15.-19.08.2011*, 2011.

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

[124] D. Littlejohn, "TRW Systems Engineering - Slip Control Systems CAR Breakfast ..." http://www.cargroup.org/wp-content/uploads/2017/02/IB_Mar2014_derron_littlejohn_presentation.pdf, 2014.

[125] F. Mc Caffery, V. Casey, M. Sivakumar, G. Coleman, P. Donnelly, and J. Burton, "Medical device software traceability," in *Software and Systems Traceability*. Springer, 2012, pp. 321–339.

[126] P. Rempel, P. Mäder, T. Kuschke, and J. Cleland-Huang, "Assessing the Conformance of Software Traceability to relevant guidelines." in *ICSE*, 2014.

[127] "Functional Design RTL – Implementation R ... - ASICentrum," http://www.asicentrum.cz/file/downloads/files/advanced_fpga_design_neil_rattray_part2.pdf.

[128] "Software Engineering ITK Engineering," http://www.itk-engineering.de/en/development-partnership-competencies/software-engineering/.

[129] G. Loniewski, E. Insfran, and S. Abrahão, "A systematic review of the use of requirements engineering techniques in model-driven development," in *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2010, pp. 213–227.

[130] "YAKINDU - Case Study - DENSO - itemis AG," https://www.itemis.com/en/references/denso/.

[131] B. Dowdeswell and E. H. R. Sinh and, "TORUS: Tracing Complex Requirements for Large Cyber-Physical Systems," in *2016 21st International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2016.

[132] M. Detmers, "Polarion Receives ISO 26262 Certification and ... - Polarion Software," https://polarion.plm.automation.siemens.com/hubfs/Docs/Press/Polarion-Software-receives-ISO-26262-IEC-61508-Qualification.pdf, 2012.

[133] "Jama Software Announces TUV SUD Certification of Its Solution for ..." http://www.marketwired.com/press-release/jama-software-announces-tuv-sud-certification-its-solution-safety-related%-development-2144922.htm, 2016.

[134] I. Galvão and A. Goknil, "Survey of traceability approaches in model-driven engineering," *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC*, pp. 313–324, 2007.

[135] J. Leuser and D. Ott, "Tackling semi-automatic trace recovery for large specifications," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010.

[136] M. Krammer, M. Karner, and A. Fuchs, "System design for enhanced forward-engineering possibilities of safety critical embedded systems," in *17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, 2014.

[137] T. Lovric, M. Schneider-Scheyer, and S. Sarkic, "SysML as backbone for engineering and safety - Practical experience with TRW braking ECU," in *SAE 2014 World Congress and Exhibition*, 2014.

[138] E.-A. Karlsson, "Traceability in an Agile process - Addalot," http://safety.addalot.se/upload/2016/PDF/1-6%20EAK-Tracebility%20through%20sprints-pub.pdf, 2016.

[139] H. Sporer and E. Brenner, "An automotive E/E system domain-specific modelling approach with various tool support," *ACM SIGAPP Applied Computing Review: Volume 16 Issue 1, March 2016*, 2016.

[140] R. Weissnegger, M. Pistauer, C. Kreiner, K. Römer, and C. Steger, "A novel design method for automotive safety- critical ... - IDA.LiU.se," in *Proceedings of the 2015 Forum on Specification & Design Languages*, 2015.

[141] A. De Lucia, F. Fasano, and R. Oliveto, "Traceability management for impact analysis," in *Frontiers of Software Maintenance, 2008. FoSM 2008.* IEEE, 2008, pp. 21–30.

[142] T. ReiÃ§, "INSTITUT FÃIJR INFORMATIK Traceability - Aktuelle ... - mediaTUM," https://mediatum.ub.tum.de/doc/1094508/1094508.pdf, 2008.

[143] "Safety and Reliability Analyzer - No Magic," https://www.nomagic.com/images/products/cameo-safety-and-reliability-analyzer/no-magic-csra-brochure-11-2016.pdf.

[144] H. Cho, "Traceability-Driven System Development and its Application to Automotive System Development," in *2014 21st Asia-Pacific Software Engineering Conference*, 2014.

[145] "Effiziente Erfassung und Pflege von Traceability ... - DepositOnce," https://depositonce.tu-berlin.de/bitstream/11303/4332/1/figge_asmus.pdf, 2014.

[146] S. Königs, G. Beie, A. Figge, and R. Stark, "Traceability in Systems Engineering - Review of industrial practices, state-of-the-art technologies and new research solutions," in *Advanced Engineering Informatics*, 2012.

[147] P. Arkley and S. Riddle, "Tailoring Traceability Information to Business Needs," in *RE'06*, 2006.

[148] CollabNet, "CollabNet TeamForge Solutions for the Automotive Industry," https://www.collab.net/sites/all/themes/collabnet/_media/pdf/ds/CollabNet_datasheet_TeamForge_Solutions_for_the_Automotive_Industry.pdf, 2013.

[149] "Softwareprototyp "Traceability-Demonstrator"," http://optrac.de/wp-content/uploads/2015/09/Traceability_Demonstrator.pdf, 2015.

[150] "www.mbtech-group.com - MBtech safeguards traceability for ..." https://www.mbtech-group.com/cz/spolecnost/novinky/news_article/article/mbtech_sichert_nachverfolgbarkeit_in_der_elektronikentwicklung.html?no_cache=1, 2009.

[151] H. Blom, H. Lönn, F. Hagl, Y. Papadopoulo, M.-O. Reiser, C.-J. Sjösted, D.-J. Chen, F. Tagliabò, S. Torchiaro, S. Tucci, and R. Kolagari, "EAST-ADL: An architecture description language for automotive software-intensive systems," in *Embedded Computing Systems: Applications, Optimization, and Advanced Design*, 2013.

[152] "Atego Trace - Products - Atego," http://www.atego.com/de/products/atego-trace/.

[153] H. Jost, A. Hahn, S. HÃďusler, S. KÃűhler, J. GaÄĐnik, F. KÃűster, and K. Lemmer, "Supporting qualification: Safety standard compliant process planning and monitoring," in *Proceedings - 2010 IEEE Symposium on Product Compliance Engineering, PSES 2010*, 2010.

[154] B. M. and F. D., "A model-based reference workflow for the development of safety-related software," in *SAE Technical Papers*, 2010.

[155] G. Gorman, "Systems with IBM IoT Continuous Engineering Solutions - National ..." ftp://ftp.ni.com/pub/branches/us/tlf/automotive/ibm_tlf_2015.pdf, 2015.

[156] I. Santiago, A. Jiménez, J. M. Vara, V. De Castro, V. A. Bollati, and E. Marcos, "Model-driven engineering as a new landscape for traceability management: A systematic literature review," *Information and Software Technology*, vol. 54, no. 12, pp. 1340–1356, 2012.

[157] "Requirements Software Automotive Industry - Visure Solutions," https://visuresolutions.com/requirements-software-automotive-industry/.

[158] "End-to-End Requirements Traceability Report –ReqView," https://www.reqview.com/blog/2016-05-31-news-end-to-end-requirements-traceability-report.html, 2016.

[159] "Traceability Matrix and Link Graph | Atlassian Marketplace," https://marketplace.atlassian.com/plugins/com.kostebekteknoloji.plugins.jira.jira-traceability-matrix, 2017.

[160] M. Broy, "A Logical Approach to Systems Engineering Artifacts and Traceability," https://pdfs.semanticscholar.org/94ad/d6aa96bee31e8acd59b534f01f3afe66134c.pdf.

[161] S. Wendler and F. Graser, "Durch Traceability lassen sich unbändige Projekte zähmen," http://www.elektronikpraxis.vogel.de/themen/embeddedsoftwareengineering/analyseentwurf/articles/431603/, 2014.

[162] "Nachverfolgbarkeit – warum eigentlich? - Systems Engineering Trends," http://se-trends.de/nachverfolgbarkeit/, 2017.

[163] B. Leigh, J. Wlad, and B. Lewis, "ISO 26262 Approval of Automotive Software Components - SlideShare," https://www.slideshare.net/RealTimeInnovations/iso-26262-approval-of-automotive-software-components, 2016.

[164] F. Corbier, M. Soodeen, S. Loembe, and G. Thurston, "Creating a systems simulation framework & roadmap," in *SAE Technical Papers*, 2013.

[165] "PLM in der Automobilzulieferindustrie | PROCAD," https://www.procad.de/blog/plm-in-der-automobilzulieferindustrie/.

[166] K. Pohl, G. Böckle, and F. J. van Der Linden, *Software product line engineering: foundations, principles and techniques.* Springer Science & Business Media, 2005.

[167] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, "Software traceability with topic modeling," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1.* ACM, 2010, pp. 95–104.

[168] P. Mäder and O. Gotel, "Towards automated traceability maintenance," *Journal of Systems and Software*, vol. 85, no. 10, pp. 2205–2227, 2012.

[169] J. Llorens, "Automatic Traceability between models and requirements," http://www.nordic-systems-engineering-tour.com/conference-program-nose-2017/automatic-traceability-between-models-and-requirements-the-trigger-for\-systems-engineering-interoperability/, 2017.

[170] A. Baumgart and C. Ellen, "A Recipe for Tool Interoperability," in *MODELSWARD 2014 - Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development*, 2014.

[171] J. Guo, J. Cleland-Huang, and B. Berenbach, "Foundations for an expert system in domain-specific traceability," in *2013 21st IEEE International Requirements Engineering Conference (RE)*, 2013.

[172] M. Borg, "In vivo evaluation of large-scale IR-based traceability recovery." in *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*, 2011.

[173] S. Weibleder, "Relation of Model-Based Testing and Safety-Relevant Standards," http://www.model-based-testing.de/mbtuc11/presentations/Weissleder-FIRST-Standards_and_MBT.pdf.

[174] "Requirements Manager (m/f) (Driver Assistance) - Job bei Elektrobit ..." http://www.stepstone.de/stellenangebote--Requirements-Manager-m-f-Driver-Assistance-Erlangen-Boeblingen-Braunschweig-Ingolstadt-Muenchen-Ulm-Elektrobit-Automotive-GmbH--4414902-inline.html.

[175] S. Software, "Managing ISO 26262 Compliance," http://downloads.seapine.com/pub/papers/managing-iso-26262-compliance-guide.pdf.

[176] A. Egyed, P. Grünbacher, M. Heindl, and S. Biffl, "Value-based requirements traceability: Lessons learned," in *Design requirements engineering: a ten-year perspective*. Springer, 2009, pp. 240–257.

[177] V. Gaur and A. Soni, "A fuzzy traceability vector model for requirements validation," *International Journal of Computer Applications in Technology*, vol. 47, no. 2-3, pp. 172–188, 2013.

[178] J. Cleland-Huang, C. K. Chang, and M. Christensen, "Event-based traceability for managing evolutionary change," *IEEE Transactions on Software Engineering*, vol. 29, no. 9, pp. 796–810, 2003.

[179] J. Cleland-Huang, G. Zemont, and W. Lukasik, "A heterogeneous solution for improving the return on investment of requirements traceability," in *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*. IEEE, 2004, pp. 230–239.

[180] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *Software Engineering, IEEE Transactions on*, vol. 25, no. 4, pp. 557–572, 1999.

[181] B. Aichernig, K. Hormaier, F. Lorber, D. Nickovic, R. Schlick, D. Simoneau, and S. Tiran, "Integration of requirements engineering and test-case generation via OSLC," *14th International Conference on Quality Software*, pp. 117–126, 2014.

[182] N. Anquetil, U. Kulesza, R. Mitschke, A. Moreira, J. C. Royer, A. Rummler, and A. Sousa, "A model-driven traceability framework for software product lines," *Software and Systems Modeling*, vol. 9, no. 4, pp. 427–451, 2010.

[183] L. Lamb, W. Jirapanthong, and A. Zisman, "Formalizing traceability relations for product lines," in *6th Int. Workshop on Traceability in Emerging Forms of Software Engineering*. ACM, 2011, p. 42.

[184] S. Walderhaug, U. Johansen, E. Stav, and J. Aagedal, "Towards a generic solution for traceability in MDD," in *ECMDA Traceability Workshop*, 2006, pp. 41–51.

[185] M. Fockel, J. Holtmann, and J. Meyer, "Semi-automatic establishment and maintenance of valid traceability in automotive development processes," *2nd Int. Workshop on Software Engineering for Embedded Systems (SEES'12)*, pp. 37–43, 2012.

[186] CMMI Product Team, "CMMI for Development, version 1.3," Software Engineering Institute, Tech. Rep. CMU/SEI-2010-TR-033, November 2010.

[187] A. Seibel, R. Hebig, and H. Giese, "Traceability in model-driven engineering: Efficient and scalable traceability maintenance," J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer London, October 2012, pp. 215–240.

[188] A. de Lucia, A. Marcus, R. Oliveto, and D. Poshyvanyk, "Information retrieval methods for automated traceability recovery," J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer London, October 2012, pp. 71–98.

[189] Z. Diskin, Y. Xiong, K. Czarnecki, H. Ehrig, F. Hermann, and F. Orejas, "From state- to delta-based bidirectional model transformations: the symmetric case," in *MODELS'11*, ser. LNCS, J. Whittle, T. Clark, and T. Kühne, Eds., vol. 6981. Springer, 2011, pp. 304–318.

[190] P. Stevens, "Bidirectionally tolerating inconsistency: Partial transformations," in *International Conference on Fundamental Approaches to Software Engineering (FASE'14)*, ser. LNCS, S. Gnesi and A. Rensink, Eds., vol. 8411. Springer, 2014, pp. 32–46.

[191] J. Cheney, J. Gibbons, J. McKinna, and P. Stevens, "Towards a principle of least surprise for bidirectional transformations," in *BX 2015*, ser. CEUR Workshop Proceedings, A. Cunha and E. Kindler, Eds., vol. 1396. CEUR-WS.org, 2015, pp. 66–80.

[192] R. Wohlrab, J.-P. Steghöfer, E. Knauss, S. Maro, and A. Anjorin, "Collaborative traceability management: Challenges and opportunities," in *Proceedings of 24th IEEE International Requirements Engineering Conference (RE' 16)*, 2016, p. 10.

[193] Z. Diskin, A. Wider, H. Gholizadeh, and K. Czarnecki, "Towards a rational taxonomy for increasingly symmetric model synchronization," in *International Conference on Theory and Practice of Model Transformations (ICMT 2014)*, ser. LNCS, D. D. Ruscio and D. Varró, Eds., vol. 8568. Springer, 2014, pp. 57–73.

[194] N. Drivalos-Matragkas, D. S. Kolovos, R. F. Paige, and K. J. Fernandes, "A state-based approach to traceability maintenance," in *Proceedings of the 6th ECMFA Traceability Workshop*. ACM, 2010, pp. 23–30.

[195] H. Schwarz, J. Ebert, and A. Winter, "Graph-based traceability: a comprehensive approach," *Software & Systems Modeling*, vol. 9, no. 4, pp. 473–492, 2010.

[196] J. Cleland-Huang, C. K. Chang, and M. Christensen, "Event-based traceability for managing evolutionary change," *Transactions on Software Engineering*, vol. 29, no. 9, pp. 796–810, 2003.

[197] I. Pete and D. Balasubramaniam, "Handling the differential evolution of software artefacts: A framework for consistency management," in *22nd Int. Conf. on Software Analysis, Evolution, and Reengineering (SANER'15)*, 2015, pp. 599–600.

[198] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *TSE*, vol. 27, no. 1, pp. 58–93, 2001.

[199] A. Egyed, S. Biffl, M. Heindl, and P. Grünbacher, "Determining the cost-quality trade-off for automated software traceability," in *20th IEEE/ACM Int. Conf. on Automated software engineering (ASE '05)*. ACM, 2005, pp. 360–363.

[200] A. D. Lucia, A. Marcus, R. Oliveto, and D. Poshyvanyk, "Information retrieval methods for automated traceability recovery," in *Software and Systems Traceability*, 2012, pp. 71–98. [Online]. Available: https://doi.org/10.1007/978-1-4471-2239-5_4

[201] J. Guo, J. Cheng, and J. Cleland-Huang, "Semantically enhanced software traceability using deep learning techniques," in *39th Int. Conf. on Software Engineering, (ICSE '17)*, 2017, pp. 3–14. [Online]. Available: https://doi.org/10.1109/ICSE.2017.9

[202] R. Michael, R. Jacob, G. Jin L.C., C. Jane, and M. Patrick, "Traceability in the wild: Automatically augmenting incomplete trace links," in *40th ACM/IEEE Int. Conf. on Software Engineering (ICSE '18)*, 2018.

[203] A. De Lucia, A. Marcus, R. Oliveto, and D. Poshyvanyk, "Information retrieval methods for automated traceability recovery," in *Software and Systems Traceability*. Springer, 2012, pp. 71–98.

[204] J. H. Hayes and A. Dekhtyar, "Humans in the traceability loop: can't live with'em, can't live without'em," in *3rd Int. Workshop on Traceability in emerging forms of software engineering (TEFSE '05)*. ACM, 2005, pp. 20–23.

[205] A. Mahmoud, "Toward an effective automated tracing process," in *20th IEEE Int. Conf. on Program Comprehension (ICPC '12)*, June 2012, pp. 269–272.

[206] N. Niu, W. Wang, and A. Gupta, "Gray links in the use of requirements traceability," in *24th Int. Symposium on Foundations of Software Engineering (FSE)*, 2016, pp. 384–395.

[207] W.-K. Kong, J. H. Hayes, A. Dekhtyar, and O. Dekhtyar, "Process improvement for traceability: A study of human fallibility," in *20th IEEE Int. Requirements Engineering Conf. (RE' 12)*. IEEE, 2012, pp. 31–40.

[208] A. Dekhtyar and M. Hilton, "Human recoverability index: a tracelab experiment," in *7th Int. Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE '13)*. IEEE, 2013, pp. 37–43.

[209] Y. Shin and J. Cleland-Huang, "A comparative evaluation of two user feedback techniques for requirements trace retrieval," in *ACM Symposium on Applied Computing, SAC 2012*, 2012, pp. 1069–1074. [Online]. Available: http://doi.acm.org/10.1145/2245276.2231943

[210] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Adams re-trace: A traceability recovery tool," in *9th European Conf. on Software Maintenance and Reengineering (CSMR '05)*. IEEE, 2005, pp. 32–41.

[211] A. Mahmoud and N. Niu, "Tracter: A tool for candidate traceability link clustering," in *19th IEEE Int. Requirements Engineering Conf. (RE' 11)*. IEEE, 2011, pp. 335–336.

[212] N. Niu, A. Mahmoud, Z. Chen, and G. Bradshaw, "Departures from optimality: Understanding human analyst's information foraging in assisted requirements tracing," in *2013 Int. Conf. on Software Engineering (ICSE '13)*, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 572–581. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486788.2486864

[213] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[214] R. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014. [Online]. Available: https://doi.org/10.1007/978-3-662-43839-8

[215] J. Cleland-Huang and M. Rahimi, "A case study: Injecting safety-critical thinking into graduate software engineering projects," in *39th IEEE/ACM Int. Conf. on Software Engineering: Software Engineering Education and Training Track, ICSE-SEET*, 2017, pp. 67–76. [Online]. Available: https://doi.org/10.1109/ICSE-SEET.2017.4

[216] M. Rahimi, W. Xiong, J. Cleland-Huang, and R. Lutz, "Diagnosing assumption problems in safety-critical products," in *32nd IEEE/ACM Int. Conf. on Automated Software Engineering (ASE '17)*. IEEE Press, 2017, pp. 473–484.

[217] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university Press, 2008, vol. 1, no. 1.

[218] G. Regan, F. McCaffery, K. McDaid, and D. Flood, "The development and validation of a traceability assessment model," in *Software Process Improvement and Capability Determination*, A. Mitasiunas, T. Rout, R. V. O'Connor, and A. Dorling, Eds. Cham: Springer International Publishing, 2014, pp. 72–83.

[219] M. P. Fay and M. A. Proschan, "Wilcoxon-mann-whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules," *Statistics surveys*, vol. 4, p. 1, 2010.

[220] S. Maro, J.-P. Steghöfer, J. Hayes, J. Cleland-Huang, and M. Staron, "Vetting automatically generated trace links: what information is useful to human analysts?" in *2018 IEEE 26th International Requirements Engineering Conference (RE)*. IEEE, 2018, pp. 52–63.

[221] O. Pedreira, F. García, N. Brisaboa, and M. Piattini, "Gamification in software engineering–a systematic mapping," *Information and Software Technology*, vol. 57, pp. 157–168, 2015.

[222] M. D. Hanus and J. Fox, "Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance," *Computers & Education*, vol. 80, pp. 152–161, 2015.

[223] J. Hamari, J. Koivisto, and H. Sarsa, "Does gamification work?–a literature review of empirical studies on gamification," in *47th Hawaii Int. Conf. on System Sciences (HICSS)*. IEEE, 2014, pp. 3025–3034.

[224] D. Charles, T. Charles, M. McNeill, D. Bustard, and M. Black, "Game-based feedback for educational multi-user virtual environments," *British Journal of Educational Technology*, vol. 42, no. 4, pp. 638–654, 2011.

[225] R. M. Parizi, "On the gamification of human-centric traceability tasks in software testing and coding," in *2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA)*. IEEE, 2016, pp. 193–200.

[226] J. H. Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval," in *11th IEEE Int. Requirements Engineering Conf. (RE' 03)*. IEEE, 2003, pp. 138–147.

[227] J. H. Hayes and A. Dekhtyar, "Humans in the traceability loop: Can't live with 'em, can't live without 'em," in *3rd Int. Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE '05)*, ser. TEFSE '05. New York, NY, USA: ACM, 2005, pp. 20–23. [Online]. Available: http://doi.acm.org/10.1145/1107656.1107661

[228] W.-K. Kong, J. Huffman Hayes, A. Dekhtyar, and J. Holden, "How do we trace requirements: an initial study of analyst behavior in trace validation tasks," in *4th Int. Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '11)*. ACM, 2011, pp. 32–39.

[229] F. Dalpiaz, R. Snijders, S. Brinkkemper, M. Hosseini, A. Shahri, and R. Ali, "Engaging the crowd of stakeholders in requirements engineering via gamification," in *Gamification*. Springer, 2017, pp. 123–135.

[230] P. Lombriser, F. Dalpiaz, G. Lucassen, and S. Brinkkemper, "Gamified requirements engineering: model and experimentation," in *REFSQ'16*. Springer, 2016, pp. 171–187.

[231] F. M. Kifetew, D. Munante, A. Perini, A. Susi, A. Siena, P. Busetta, and D. Valerio, "Gamifying collaborative prioritization: Does pointsification work?" in *RE'17*. IEEE, 2017, pp. 322–331.

[232] M. Z. H. Kolpondinos and M. Glinz, "Behind points and levelsâĂŤthe influence of gamification algorithms on requirements prioritization," in *RE'17*. IEEE, 2017, pp. 332–341.

[233] D. L. Kappen and L. E. Nacke, "The kaleidoscope of effective gamification: deconstructing gamification in business applications," in *Proc. of the 1st. Int. Conf. on Gameful Design, Research, and Applications*. ACM, 2013, pp. 119–122.

[234] B. Morschheuser, J. Hamari, K. Werder, and J. Abe, "How to gamify? a method for designing gamification," 2017.

[235] F. García, O. Pedreira, M. Piattini, A. Cerdeira-Pena, and M. Penabad, "A framework for gamification in software engineering," *Journal of Systems and Software*, vol. 132, pp. 21–40, 2017.

[236] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[237] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press, 2008, vol. 39.

[238] S. Maro, E. Sundklev, C.-O. Persson, G. Liebel, and J.-P. Steghöfer, "Impact of Gamification on Trace Link Vetting: a Controlled Experiment," Jan. 2019, Dataset. [Online]. Available: https://doi.org/10.5281/zenodo.2540646

[239] D. J. Dubois and G. Tamburrelli, "Understanding gamification mechanisms for software development," in *FSE'13*. ACM, 2013, pp. 659–662.

[240] A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," *Journal of Usability Studies*, vol. 4, no. 3, pp. 114–123, 2009.

[241] C. Eickhoff, C. G. Harris, A. P. de Vries, and P. Srinivasan, "Quality through flow and immersion: gamifying crowdsourced relevance assessments," in *Proc. of the 35th Int. ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2012, pp. 871–880.

[242] E. D. Mekler, F. Brühlmann, K. Opwis, and A. N. Tuch, "Do points, levels and leaderboards harm intrinsic motivation?: an empirical analysis of common gamification elements," in *Proc. of the 1st Int. Conf. on Gameful Design, Research, and Applications*. ACM, 2013, pp. 66–73.

[243] E. L. Deci, R. Koestner, and R. M. Ryan, "A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation." *Psychological bulletin*, vol. 125, no. 6, p. 627, 1999.

[244] P. Jönsson and M. Lindvall, "Impact analysis," in *Engineering and managing software requirements*. Springer, 2005, pp. 117–142.

[245] P. Rempel, P. Mäder, T. Kuschke, and J. Cleland-Huang, "Mind the gap: assessing the conformance of software traceability to relevant guidelines," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 943–954.

[246] P. Rempel, P. Mäder, and T. Kuschke, "An empirical study on project-specific traceability strategies," in *Requirements Engineering Conference (RE), 2013 21st IEEE International*. IEEE, 2013, pp. 195–204.

[247] J. Cleland-Huang, J. H. Hayes, and J. M. Domel, "Model-based traceability," in *2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*. IEEE, 2009, pp. 6–10.

[248] M. C. Panis, "Successful deployment of requirements traceability in a commercial engineering organization... really," in *Requirements Engineering Conference (RE), 2010 18th IEEE International*. IEEE, 2010, pp. 303–307.

[249] A. Mahmoud and N. Niu, "Supporting requirements traceability through refactoring," in *2013 21st IEEE International Requirements Engineering Conference (RE)*. IEEE, 2013, pp. 32–41.

[250] T. Wolfenstetter, M. R. Basirati, M. Böhm, and H. Krcmar, "Introducing trails: A tool supporting traceability, integration and visualisation of engineering knowledge for product service systems development," *Journal of Systems and Software*, vol. 144, pp. 342–355, 2018.

[251] M. Mezghani, J. Kang, E.-B. Kang, and F. Sedes, "Clustering for traceability managing in system specifications," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 2019, pp. 257–264.

[252] W. Engelsman, R. J. Wieringa, M. van Sinderen, J. Gordijn, and T. Haaker, "Realizing traceability from the business model to enterprise architecture," in *International Conference on Conceptual Modeling*. Springer, 2019, pp. 37–46.

[253] J. M. Florez, "Automated fine-grained requirements-to-code traceability link recovery," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2019, pp. 222–225.

[254] J. M. Vara, V. A. Bollati, Á. Jiménez, and E. Marcos, "Dealing with traceability in the mddof model transformations," *IEEE Transactions on Software Engineering*, vol. 40, no. 6, pp. 555–583, 2014.

[255] D. Ståhl, K. Hallén, and J. Bosch, "Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework," *Empirical Software Engineering*, vol. 22, no. 3, pp. 967–995, 2017.

[256] S. Nair, J. L. de la Vara, A. Melzi, G. Tagliaferri, L. De-La-Beaujardiere, and F. Belmonte, "Safety evidence traceability: Problem analysis and model," in *International working conference on requirements engineering: Foundation for software quality*. Springer, 2014, pp. 309–324.

[257] B. Wang, R. Peng, Y. Li, H. Lai, and Z. Wang, "Requirements traceability technologies and technology transfer decision support: A systematic review," *Journal of Systems and Software*, vol. 146, pp. 59–79, 2018.

[258] P. Mäder and O. Gotel, "Ready-to-use traceability on evolving projects," in *Software and Systems Traceability*. Springer, 2012, pp. 173–194.

[259] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[260] 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)*. IEEE, 2016, pp. 216–225.

[261] J.-P. Steghöfer, "Software traceability tools: Overview and categorisation," in *Report of the GI Working Group "Traceability/Evolution"*. German Informatics Society (GI), October 2017, pp. 2–7. [Online]. Available: http://pi.informatik.uni-siegen.de/gi/stt/38_1/01_Fachgruppenberichte/ARC_AKTE/ARC_AKTE_2017_p2_steghoefer.pdf

[262] J. T. Biehl, M. Czerwinski, M. Czerwinski, G. Smith, and G. G. Robertson, "Fastdash: a visual dashboard for fostering awareness in software teams," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2007, pp. 1313–1322.

[263] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *2011 International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2011, pp. 275–284.

[264] B. Pages, "Bouml," 2018, accessed on: 23-05-2019. [Online]. Available: https://www.bouml.fr/index.html

[265] F. Kinoshita, "Practices of an agile team," in *Agile 2008 Conference*. IEEE, 2008, pp. 373–377.

[266] Itemis, "Yakindu traceability," 2019, accessed on: 07-08-2019. [Online]. Available: https://www.itemis.com/en/yakindu/traceability/

[267] P. Pruski, S. Lohar, W. Goss, A. Rasin, and J. Cleland-Huang, "Tiqi: answering unstructured natural language trace queries," *Requirements Engineering*, vol. 20, no. 3, pp. 215–232, 2015.

[268] P. Mäder and J. Cleland-Huang, "A visual language for modeling and executing traceability queries," *Software & Systems Modeling*, vol. 12, no. 3, pp. 537–553, 2013.

[269] M. Staron, "Adopting model driven software development in industry – a case study at two companies," in *Model Driven Engineering Languages and Systems*, O. Nierstrasz, J. Whittle, D. Harel, and G. Reggio, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 57–72.

[270] J. P. Kotter and D. S. Cohen, *The heart of change: Real-life stories of how people change their organizations.* Harvard Business Press, 2002.

[271] J. Cleland-Huang, O. Gotel, and A. Zisman, Eds., *Software and Systems Traceability.* Springer London, October 2011.