Requirements Engineering
https://doi.org/10.1007/s00766-021-00361-5

ORIGINAL ARTICLE

TracIMo: a traceability introduction methodology and its evaluation in an Agile development team

Salome Maro[1] · Jan-Philipp Steghöfer[2,5] · Paolo Bozzelli[3] · Henry Muccini[4]

Received: 5 September 2019 / Accepted: 17 August 2021

Abstract
Software traceability, the ability to relate software development artifacts such as requirements, design models and code to each other, is an important aspect in software development. It 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—a plan for how traceability should be managed—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 a Traceability Introduction Methodology (TracIMo), which is a methodology for systematically designing, implementing and evaluating software traceability in practice. We used design science research to design TracIMo and evaluated it in a case study with 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. We also report practical challenges encountered when designing a traceability strategy such as defining the right level of granularity and the need for defining intermediate development artifacts.

keywords Traceability · Software traceability · Traceability management

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" [12]. 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 [14, 30], program comprehension [5] and compliance to standards [57].

Even with all the promised benefits, many companies developing software lack systematic traceability strategies [37]. Trace links are created and maintained in an ad hoc manner and therefore benefits are not visible due to the mismatch between the established strategy and the traceability needs of the company [56]. A traceability strategy is a plan of action for how traceability should be established and maintained in an organization. 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 [23].

One of the reasons for ad hoc traceability is the lack of concrete guidelines for practitioners on how to establish traceability [11, 37, 43]. This can lead to effort invested in creating and maintaining trace links which are ultimately inconsistent, incomplete and never used [9]. While there is literature reporting on case studies in which traceability

✉ Salome Maro
maro.salome@udsm.ac.tz

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

Paolo Bozzelli
paolo.bozzelli@gmail.com

Henry Muccini
henry.muccini@univaq.it

1 University of Dar es Salaam, Dar es Salaam, Tanzania
2 Chalmers | University of Gothenburg, Gothenburg, Sweden
3 Knab, Amsterdam, The Netherlands
4 University of L'Aquila, L'Aquila, Italy
5 Chalmers, Gothenburg, Sweden

Table 1 Publication venues and the number of papers on software traceability

Venue	Search string	No. of papers
Requirements Engineering Conference (RE)	TITLE("Traceability") AND CONFNAME("Requirements Engineering").	77
Requirements Engineering Foundations for Software Quality Conference (REFSQ)	TITLE("Traceability") AND CONFNAME("REFSQ")	15
International Conference of Software Engineering Conference (ICSE)	TITLE("Traceability") AND CONFNAME("ICSE")	36
Automated Software Engineering (ASE)	TITLE("Traceability") AND CONFNAME("ASE")	27
Foundations of Software Engineering Conference (FSE)	TITLE("Traceability") AND CONFNAME("FSE")	9
Transactions of Software Engineering Journal (TSE)	TITLE("Traceability") AND ISSN (0098-5589)	9
Journal of Software and Systems (JSS)	TITLE("Traceability") AND ISSN (0164-1212)	17
Information Software Technology Journal (IST)	TITLE("Traceability") AND ISSN (0950-5849)	15
Requirements Engineering Journal (REEN)	TITLE("Traceability") AND ISSN (0947-3602)	4
ACM Transactions of Software Engineering and Methodology (TOSEM)	TITLE("Traceability") AND ISSN (1049-331X)	2
Empirical Software Engineering Journal (EMSE)	TITLE("Traceability") AND ISSN (1382-3256)	9

is established (see, e.g., [3, 49]), 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. [15], 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 [11]. An attempt toward addressing a related, but different problem is the study by Rempel et al. [56], 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 pursue the following research goal:

RG: Provide support for establishing a traceability strategy that allows the organization to achieve its goals and measure the impact of the traceability strategy.

The contribution of this paper is threefold: first, we present TracIMo, a structured methodology for designing a traceability strategy and introducing traceability in software development organizations; second, we describe in detail how we used TracIMo to design a traceability strategy in a concrete organization; 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 Sect. 2 we discuss previous studies related to our work and compare them to TracIMo. Section 3 describes our research methodology. In Sect. 4, we describe Traceability Introduction Methodology (TracIMo), while Sect. 5 describes how we applied and evaluated TracIMo at a company. Section 6 provides a discussion with respect to the research goal. Section 7 describes the threats to validity of our study, and Sect. 8 concludes the paper and outlines future work.

2 Related work

A vast amount of traceability research covering various topics is available. We performed two distinct steps when looking for existing approaches to establish traceability in an organization. First, we started with related work on traceability strategies that we were familiar with and performed an opportunistic literature search using a variety of search strings in Google Scholar, IEEE Xplore, Scopus, and the ACM Digital Library to identify further studies. Second, to ensure that we did not miss any important papers, we performed a lightweight systematic mapping study by searching the eleven top software engineering publication venues where traceability research is usually published (see Table 1) using SCOPUS. To be as broad as possible, we searched for papers that specifically mentioned the term "traceability" in the title. For conferences, we used conference names or abbreviations, and for journals we used the ISSN number as shown with the specific search strings in Table 1.

The search was performed in mid-2020, and we did not restrict the publication time of the papers. We also allowed for inclusion of workshop papers in order to get a full understanding of the research conducted in the traceability area. An overview of the papers we found is included in our online supplementary materials [2]. 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. *We did not find a paper that provided a readily applicable framework or methodology.* Our 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., [39, 69]), automation of trace link creation using information retrieval and machine learning approaches (e.g. [10, 25, 44]), traceability between specific artifact types, e.g., business models to architecture [17] or requirements to code [19], as well as research that leverages model-based development techniques for traceability, e.g., [9, 63]. A large amount of this research is evaluated using example systems from universities (e.g., [69]) or using example systems taken from industry which are not publicly available (e.g., [25]). There are only few studies where approaches are evaluated on running industrial projects (e.g., [59]).

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 [11]. 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. [46] 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 information model (TIM) which describes the artifacts and trace link types needed for safety evidence traceability. Rempel et al. [57] 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 [7, 45] and [65] on traceability as well as overview papers on traceability research such as [11] also support our observations. For instance, [43, 45] and [54] 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 below in Sect. 2.1 and compare them with the needs we are addressing in this paper. 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 Sect. 2.3.

2.1 Frameworks for designing traceability strategies

The study by Rempel et al. [56] 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 organization'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. [23] 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 [15] 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.

Table 2 An analysis of how TracIMo compares to existing frameworks for defining traceability strategies

Characteristics	Rempel et al. [56]	Gotel et al. [23]	Dömges and Pohl [15]	Espinoza and Garbajosa [18]	Mäder et al. [36]	TracIMo
Provides process guidance	✗	✓	✓	✓	✓	✓
Provides guidance on TIMs	✓	✓	✓	✓	✓	✓
Allows assessment of existing traceability strategies	✓	✓	✓	✗	✗	✓
Provides guidance on measurement design	✗	✓	✗	✗	✗	✓
Provides guidance on tool selection	✗	✗	✗	✗	✗	✓
Provides guidance on process deployment	✗	✗	✓	✓	✗	✓
Describes concrete steps in each activity	✓	✗	✗	✓	✓	✓

The ✓mark indicates that the methodology fulfills the respective characteristic while the ✗mark indicates the opposite

Espinoza and Garbajosa [18] 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 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 [36] 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.

2.2 A comparison of TracIMo with existing works

From our review of the related work and the discussions between the researchers and collaborators from the company where the case study was conducted, we defined six criteria that we used to compare TracIMo to the existing frameworks or methodologies presented above:

1. Allows designing a traceability strategy. For this, we check if the methodology facilitates designing a traceability strategy either by providing the steps which need to be carried out or proposing how the traceability information model should be created. This criterion is therefore divided into two sub-criteria: provides guidance on the definition of the traceability process; and provides guidance on the definition of a traceability information model.
2. Allows the assessment of the existing traceability strategy. In this criterion, we check if the proposed methodology includes steps for analyzing the existing traceability strategy for improvement and alignment purposes. This is inspired by Rempel et al. [56] and work in software process improvement which emphasizes the need for assessing process changes (see, e.g., [16]).
3. Provides guidance on measurement design. For this criterion, we analyze if the framework gives any guidance on how to define concrete and customized quantitative and qualitative measurements for the existing traceability strategy. Again, this is inspired by work in the software process improvement area, e.g., [16, 62].
4. Provides guidance for tool selection. We added this criterion that assesses if the methodology provides any information on how to select suitable traceability tools based on the reported difficulty to select and customize traceability tools in industrial practice (see, e.g., [33, 41]).
5. Provides guidance on deployment of traceability, where we check if the methodology provides information and guidance on how to roll out the defined traceability strategy. This is motivated by the reported difficulties in deploying process improvement initiatives and ensure long-term adoption, e.g., in [47].
6. Describes concrete steps in each activity involved in designing, deployment and evaluation of traceability strategies. We added this as an extra characteristic to check the level of detail provided by the methodology since this is useful if practitioners want to apply the methodology in question. We therefore assess if the methodology provides detailed and concrete description of how the steps and guidance provided need to be carried out.

As depicted in Table 2, TracIMo takes inspiration from the existing proposed methodologies and addresses the gaps that these methodologies do not cover. A missing aspect in many of them is the definition of metrics to understand if the

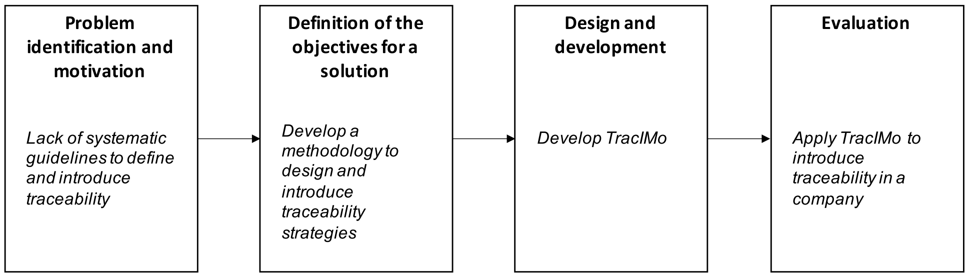
Problem identification and motivation
Lack of systematic guidelines to define and introduce traceability
Definition of the objectives for a solution
Develop a methodology to design and introduce traceability strategies
Design and development
Develop TracIMo
Evaluation
Apply TracIMo to introduce traceability in a company

Fig. 1 The design science process we followed for this contribution. Adapted from [50]

defined strategy works. While some methodologies briefly mention that the strategy needs to be continuously assessed and improved, there is no guidance on how to go about this. TracIMo covers this gap by proposing the use of GQM as well as recommending steps for when to collect the metrics. The application of TracIMo at the case company also shows examples of metrics in practice. Additionally, the level of detail in the existing methodologies is quite low. This leaves practitioners with questions on how to concretely perform the different steps required to define a traceability strategy. TracIMo covers this gap by proposing concrete activities and examples in each of the required steps.

2.3 Case studies on introducing traceability

Arkley and Riddle [3] 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, like ours, 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. [4] 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 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 [49], describes 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 in everyday tasks of developers and not a separate report.

Stål et al. [59] 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.

Amalfitano et al. [1] report their experiences on using tool integration to support traceability specifically for the testing process. In this study, the authors show three steps in which they used to design the tool integration solution which involves analyzing the existing development processes and tools before designing the tool integration solution.

In summary, there are very few studies on the introduction of traceability. This 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.*

3 Research method

In this study, we used design science [66] 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 [66]. 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. [50]: (1) problem identification and motivation; (2) definition of the objectives for a solution; (3) design and development; and (4) evaluation. These activities are described in the next subsections. Figure 1 shows the design science research methodology process of our study.

3.1 Problem identification and motivation

This first step in design science is to understand 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 organization'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 expertize 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. [38] and Maro et al. [43]. 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 [21] (see also Sect. 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 life cycle.

Our review of existing literature also showed that no methodology for the introduction of traceability existed to solve the issue. We thus formulated our problem and thus our *problem* as follows:

RP: There is a lack of systematic guidelines to define and introduce a traceability strategy.

3.2 Definition of the objectives for a solution

From previous work on software traceability (e.g., [15, 37, 41, 43, 55, 68] as discussed in Sect. 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 [23]. Our *objective* or research goal is therefore:

RG: Provide support for establishing a traceability strategy that allows the organization to achieve its goals and measure the impact of the traceability strategy.

We achieve this by defining TracIMo, a methodology for systematically designing and introducing traceability strategies.

3.3 Design and development

In this step, we developed TracIMo itself. We (the researchers together with the BA at the company) used the existing studies described in Section 2 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. [57] 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 of 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 organization over a longer period of time. Importantly, we added steps for adapting tools, deploying the strategy, and measuring its effects in the organization.

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 evaluated by designing, introducing, and assessing a traceability strategy at the company.

3.4 Evaluation

In this step, we evaluate the applicability of the designed artifact (TracIMo). This was done through a case study described in Sect. 5, where TracIMo was used to design, deploy and evaluate a traceability strategy for the development team in the company we collaborated with. This evaluation acts as proof of concept on how TracIMo can be applied in practice as well as shows areas for improvement of TracIMo. Case studies are a valid and often used tool to evaluate the artifact developed in design science research [50, 51, 64].

3.4.1 The case and context

The case is an agile development team of a company in the finance domain which wishes to introduce traceability. We worked with the IT department development team (*unit of analysis*) which consists of roles such as business analyst, lead developer, and other roles as necessary. The researchers applied the different steps of TracIMo to define the strategy and refined them in several iterations. Further details of the company are described in Section .

3.4.2 Data collection

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 as described in detail in Sects. 5.1– 5.6. The designed traceability strategy was deployed at the company. Data were collected for evaluation of the strategy in three iterations (see Sect. 5.7). The first iteration was conducted in the same week as the traceability strategy was deployed. We conducted one semi-structured interview with the product owner (PO) 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 BA, lead developer and two front-end developers. 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; 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 and the number of tickets planned per sprint to understand how accurate the development team was at effort estimation since the measurement plan required this data.

3.4.3 Data analysis

Thematic coding was used 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 were analyzed according to the metrics defined using TracIMo.

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. 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 [56], Fig. 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.

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.

4.1.1 Steps 1 and 2—Analyze 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, they are presented together.

Activities The main activities in these steps are:

1. *Collect data on the development process and traceability goals* These 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 supplemental material [2]. While observations and document analysis are good ways to determine the status quo within the organization, interviews are the main source of information about practiced process and traceability goals.

2. *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 analyze 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[1] or Essence[2] 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 organization 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 organization 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 organization.

   As an addition to the Rempel et al. framework, TracIMo uses the Goal/Question/Metric (GQM) approach [62] to achieve a standardized 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."
3. *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 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.
4. *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. [8], 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:

1. a conceptual model of the process, including roles, activities, artefacts, and tools that are used in the development process;
2. the process goals along with their rationales;
3. the traceability goals along with their rationales;
4. traceability metrics and a measurement plan; and
5. exemplary traceability scenarios.

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

[1] https://www.omg.org/spec/SPEM/About-SPEM/.

[2] https://www.omg.org/spec/Essence/About-Essence/.

carry additional meta-data, such as when it was created or who created it.

Input The inputs to this step are:

1. the process model from Step 1;
2. the traceability goals from Step 2; and
3. the traceability scenarios from Step 2.

Activities To derive the TIM, the following activities should be conducted:

1. *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 with Rempel et al. [56]. 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.
2. *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 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.
3. *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).

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 process goals that require traceability are covered by at least one traceability goal.

Input The required inputs are:

1. the process goals from Step 1;
2. the traceability goals from Step 2;

Activities To assess the traceability goals w.r.t. the process goals, the following activities should be performed:

1. *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.
2. *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 organization 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 has 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:

1. an assessment report of the process goals against the traceability goals, represented as a table relating the process goals and the corresponding traceability goals;

2. a list of refined process and/or traceability goals (optional).

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:

1. the traceability goals from Step 2;
2. the traceability scenarios from Step 2; and
3. the TIM from Step 3.

Activities To assess the TIM w.r.t. the traceability goals, the following steps should be taken:

1. *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.
2. *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 alignment is achieved. 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:

1. 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
2. a list of exemplary trace links created for specific traceability scenarios.

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:

1. the process model from Step 1;
2. the traceability goals and associated metrics from Step 2;
3. the traceability scenarios from Step 2;
4. the TIM from Step 3.

Activities To derive the traceability process the following activities are conducted:

1. *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.
2. *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.
3. *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.
4. *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.
5. *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).

6. *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.
7. *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.

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.

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:

1. the process model defined in Step 1;
2. the TIM defined in Step 3; and
3. the traceability process defined in Step 6.

Activities The following activities are conducted in this step:

1. *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.
2. *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. [55], e.g., gives an overview of steps to elicit tool requirements and important factors to consider. Additionally, Gotel and Mäder [20] provide characteristics that can be used to compare different traceability tools. The latest work is a study by Steghöfer [61] 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. [41] 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 categorization defined by Steghöfer [61] or Rempel et al. [55]. 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.
3. *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 customizations 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:

1. 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;
2. a customized traceability tool or an off-the-shelf traceability tool or an in-house developed tool.

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 organization. TracIMo recommends to deploy the process incrementally, i.e., one project at a time.

Input The inputs of this step are:

1. the traceability process from Step 6; and
2. the customized traceability tool from Step 7.

Activities The following activities are required for the deployment:

1. *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.
2. *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.
3. *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.
4. *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.
5. *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.
6. *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:

1. a deployment plan describing the concrete steps and their timing to introduce traceability tools, practices, and training;
2. baseline measurements before the roll-out of the traceability process; and
3. a deployed traceability strategy used by the involved stakeholders.

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:

1. baseline measurements before the roll-out of the traceability process from Step 8; and
2. 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:

1. *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.

2. *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.

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 organization. This step requires that the traceability strategy is deployed and used in the organization.

Activities TracIMo recommends the following activities to ensure that the deployed process is anchored:

1. *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.
2. *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.
3. *Include traceability metrics in dashboards* Many development teams use dashboards (see, e.g., [6]) to visualize 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 visualized this way to incentivize stakeholders to stick to traceability practices.

Such steps often require a more formalized 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:

1. updated training material for the development process and the traceability strategy;
2. guidelines for including traceability in reviews; and
3. automated measurement of relevant data for traceability metrics and inclusion in dashboards.

5 Case study to evaluate TracIMo in a company

To evaluate TracIMo, we applied it in a company in the finance domain. 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: 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 is 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.

The following subsections describe how TracIMo was applied to introduce traceability in the company. Some of the steps (e.g., Step 1 and Step 2) use the same data and were therefore carried out in parallel.

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 in the supplemental material [2]. Both interviews were recorded, transcribed and analyzed. We used the thematic coding approach [13] 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 3. Table 3 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 [2]. The researchers investigated each goal and proposed a number of possible metrics. These metrics were analyzed for feasibility together with the BA to see if the data needed to evaluate the metrics are 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 3.

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 in Table 5; and, (2) traceability goals which are summarized in Table 4. Additionally, the traceability goals include questions, metrics and scenarios as exemplified in Table 3. 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: PO, 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. Fig. 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 PO 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 POs 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.

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 is 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 3. 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 Fig. 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 4 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 Fig. 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.

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 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 5.

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 PO 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 PO, will be collected by the BA.

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 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 [61] where the authors have analyzed 23 existing traceability tools, to select a tool to use. This categorization 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 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 shows the six characteristics and possible values. We disregarded commercial tools and remained with five tools whose categorization is shown in Table 8.

Based on the tool assessment and the requirements from the company, we selected Eclipse Capra [40] 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.

The fact that the researchers are familiar with the customization of the tool was probably the most relevant. Since Eclipse Capra is based on the popular Eclipse IDE,[3] it requires the use of this development environment. Additionally, both the BA and the lead developer 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 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.

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 in document (2) in the supplemental material [2]) 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.

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

[3] http://www.eclipse.org.

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 [48]. The resulting UML models were imported into Papyrus[4] 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 Fig. 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 Fig. 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 Sect. 5.7.

5.7 Step 9: Evaluate process and tool

TracIMo suggests to continuously evaluate the developed traeability strategy. We implemented three evaluation phases: one initial evaluation during and immediately following deployment, one follow-up evaluation after two weeks, and a final evaluation five months after deployment.

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 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, thus adapting the traceability strategy early.

To further evaluate the deployed process and tool, we also interviewed the PO, 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 as part of the supplemental material [2]. Additionally, we conducted a focus group meeting with the BA, two front-end developers and the lead developer to discuss and prioritize the previously elicited goals and how the new process would help achieve them. Based on the feedback collected from the interview with the PO 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 using Steps 1 and 2 of TracIMo in Sect. 5.1. 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 analyzed.

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 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 of the traceability strategy defined using TracIMo:

[4] https://www.eclipse.org/papyrus/.

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 9 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.

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 9). 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.

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.

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 step is very important to ensure long-term benefits of traceability in the company and to develop the capabilities within the organization, 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.

6 Discussion

In this section, we discuss the results of the study with respect to the research goal stated in Sect. 1:

RG: Provide support for establishing a traceability strategy that allows the organization to achieve its goals and measure the impact of the traceability strategy.

This research goal indicates that there are two aspects we needed to address: provide guidance for designing a traceability strategy to fill a gap for researchers and practitioners (again, we refer to, e.g., [11, 37, 43]) and to include metrics and measurements into the strategy to allow measurement of potential benefits and drawbacks as indicated by literature on software process improvement (as discussed in Section and in, e.g., [16, 56]). We designed TracIMo to include both aspects and demonstrated with the case study whose results we reported in the previous section that TracIMo serves these purposes.

In this section, we discuss key points w.r.t. to designing a traceability strategy in Sect. 6.1 and measuring its impact in Sect. 6.2. Additionally, we encountered several challenges as a result of the traceability strategy we designed. These challenges are discussed in Sect. 6.3.

6.1 Designing a tailored traceability strategy

The study proposes TracIMo, a methodology to define a traceability strategy for software development organizations. The steps in this methodology (cf. Fig. 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. They also provide the opportunity to define metrics that allow measuring the impact a traceability strategy defined with TracIMo has. Simpler versions of steps 1 to 5 of the framework have already been shown to be effective in practice [56] 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 [62], an established technique from software process improvement, we were able to achieve both aspects. As shown in Tables 4 and 5, the thorough analysis proposed

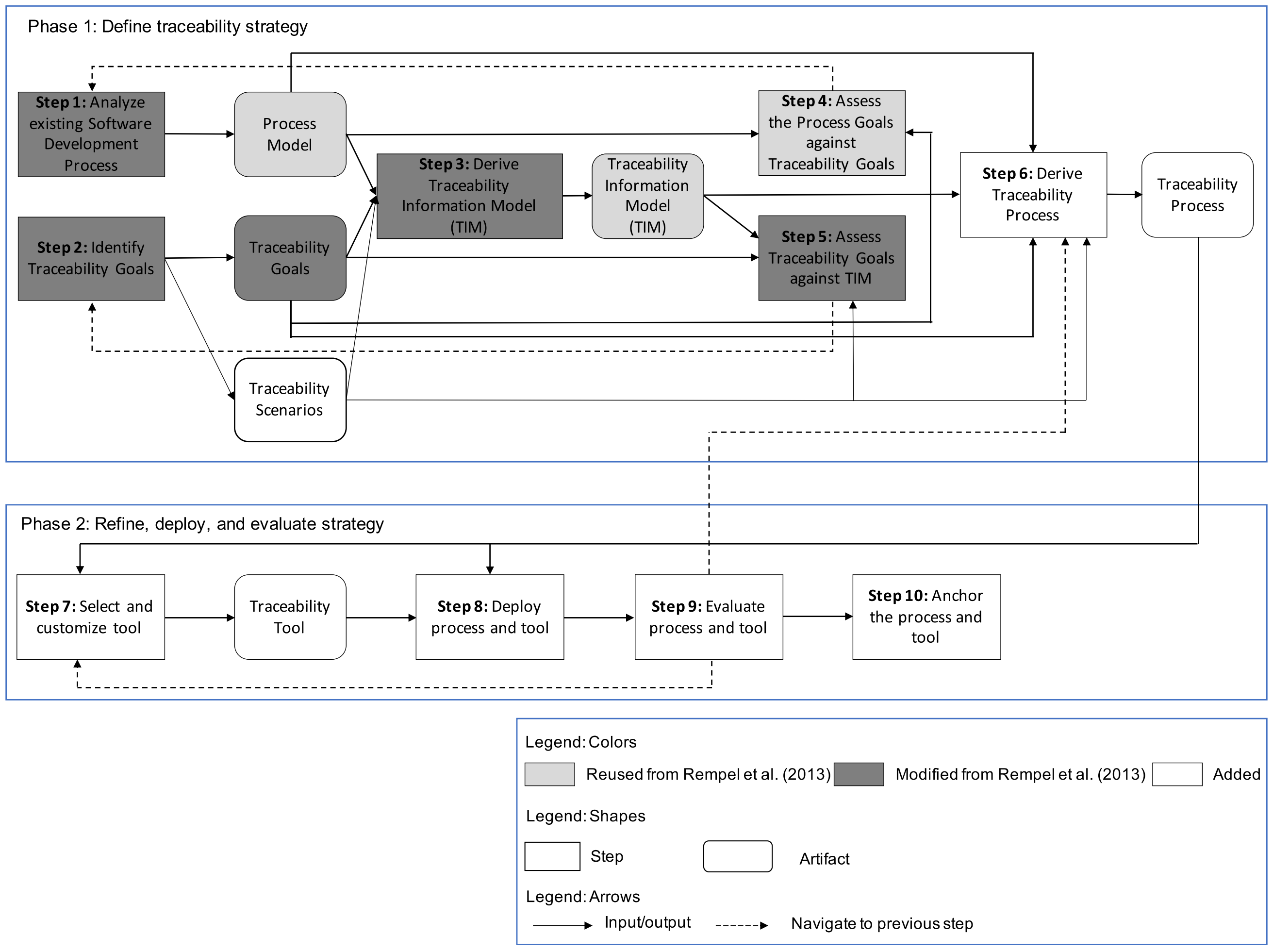
Phase 1: Define traceability strategy
Step 1: Analyze existing Software Development Process
Process Model
Step 2: Identify Traceability Goals
Traceability Goals
Step 3: Derive Traceability Information Model (TIM)
Traceability Information Model (TIM)
Step 4: Assess the Process Goals against Traceability Goals
Step 5: Assess Traceability Goals against TIM
Step 6: Derive Traceability Process
Traceability Process
Traceability Scenarios
Phase 2: Refine, deploy, and evaluate strategy
Step 7: Select and customize tool
Traceability Tool
Step 8: Deploy process and tool
Step 9: Evaluate process and tool
Step 10: Anchor the process and tool
Legend: Colors
Reused from Rempel et al. (2013)
Modified from Rempel et al. (2013)
Added
Legend: Shapes
Step
Artifact
Legend: Arrows
Input/output
Navigate to previous step

Fig. 2 Schematic overview of TracIMo, indicating which steps have been reused or modified from Rempel et al. [56] and which were added. The dashed lines represent going back to previous steps for refinement since TracIMo is iterative

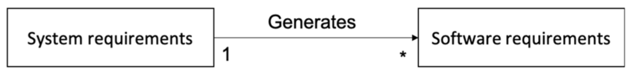

Fig. 3 An example of a simple TIM with one trace link type (generates) which connects system requirements to software requirements

by TracIMo allowed us to define a traceability information model that is specific for the company. TracIMo also ensured the traceability goals and process goals are compatible. In communicating with the company, the clearly defined goals allowed us to discuss the scope of the changes as well as what is realistically achievable with traceability and gave us a way to evaluate the defined traceability strategy.

TracIMo also exploits GQM's strengths by defining metrics that allow us to measure the benefits of the defined traceability strategy. Each traceability goal was associated with a number of metrics. In our evaluation case study, we collected data as a baseline before the introduction of traceability for some of them. This provision in TracIMo thus allowed us to compare the baseline with data after traceability was introduced. In our concrete case, advantages could be identified as shown in Table 9.

The iterations built into phase 2 of TracIMo also proved helpful. This is particularly evident in the evolution of the traceability information model. In the case study, we discovered the issues with the TIM only through deployment and evaluation of traceability in practice. Since TracIMo is 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 agile team's needs. Several studies discuss traceability for plan-driven development processes (e.g. [3, 4, 53]), 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

Table 3 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: Metrics	How much does the estimated effort differ from the actual effort?
	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: Metrics	How confident is the lead developer in the estimation of tasks?
	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

created from requirements to other development artifacts like design models and code [24, 67]. 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 [31]. We tailored the traceability process to the development process of the company by defining a *ticket-centric traceability strategy*.

As shown in Fig. 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 ticket-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.

In summary, the case study shows that the steps in TracIMoare necessary and sufficient to create a viable traceability strategy that is aligned with the current development process and includes the ability to measure its benefits. We therefore consider the case study proof that our research goal has been achieved.

6.2 Measuring the impact of the traceability strategy

As part of the steps of TracIMo, we defined metrics and a plan how to collect the measurements to evaluate the new traceability strategy in the context of the existing process. In the following, we are going to discuss some of the challenges when putting this part of TracIMo into practice in our evaluative case study. While the details will differ in other strategies designed with TracIMo, we still believe that these insights can provide additional insights when applying the methodology in practice.

Table 4 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

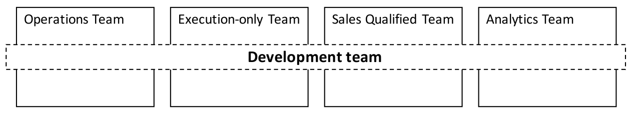

Fig. 4 Organization structure of the company

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 Sect. 5.

As discussed in [27], 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.

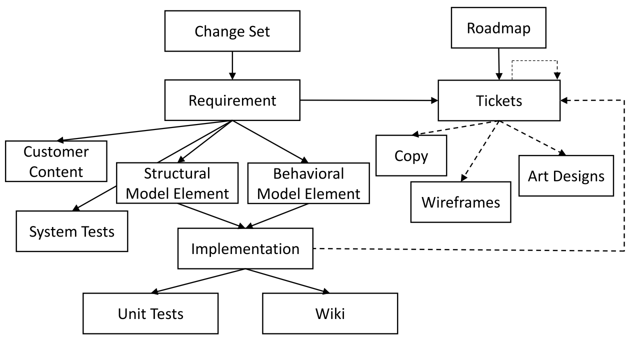

Fig. 5 Initial traceability information model. Links shown as dotted lines were already captured at the company

Table 5 Assessment of Process Goals against Traceability Goals (cf. Table 4)

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 PO/BA's point of view	
	Process Goal 2 Allow breakdown of all requirements into actionable tasks from a PO/BA's point of view	
	Process Goal 3 Improve the identification of related requirements from the PO/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	Supported by *TG 2* and achieved using a similar reasoning as for Process Goal 3
	Process Goal 12 Allow verifying features from a tester's point of view	Supported by *Traceability Goal 4* and achieved using a similar reasoning as for Process Goal 8
	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	

Table 5 (continued)

Current practice	Process goal	Support by traceability goals (TG)
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	Supported by *TG 2* and achieved using a similar reasoning as for Process Goal 3
	Process Goal 15 Improve effort estimation of requirements from a PO/BA's point of view	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
	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	

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

Table 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)

Having said this, one of the major challenges of traceability is the inability to measure its RoI [22, 54]. 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 [27]. It is also difficult to determine the entire cost of traceability in the development life-cycle [22]. One of our long-term goals, independent of TracIMo, was to investigate the RoI of traceability for the company. However, due to organizational changes that occurred at the company, this data collection was not 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, these kinds of measurements do not allow to quantify the RoI after a short time. We did, however, 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 and plannable, we provide our recommendations on how to effectively apply TracIMo in Sect. 6.4.

6.3 Challenges of traceability

While TracIMo contains explicit steps for, e.g., the definition of a traceability information model, the concrete form of such artefacts is based on many factors. We encountered five challenges as well as important decisions that needed to be made during the design of the traceability strategy with TracIMo that we believe can be encountered in other cases as well:

1. trace link granularity;
2. scope of the trace links;
3. the need for intermediate artifacts;
4. time required to create links; and
5. adoption of the traceability process.

While these challenges are not new to the research community, we discuss how they manifested in the case study and how we dealt with them in order to provide additional practical insights for both practitioners and researchers.

Table 7 Characteristics used to assess the tool and possible values

Characteristic	Possible value
Information storage	Centralized, Distributed, Separate Model, Inline
Level of integration	Holistic, Hybrid, Separate
Tool type	Application Life cycle 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, Visualization
Automation	Link generation, Consistency checking, Workflow enforcement

Table 8 Assessment of traceability tools [61]

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)	Centralized	Hybrid	Framework (Eclipse)	Reporting	Unknown

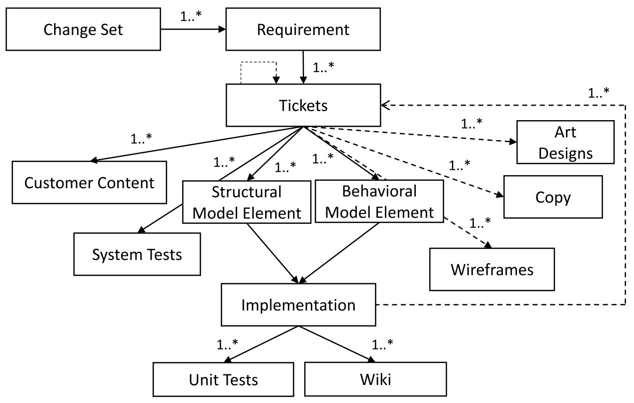

Fig. 6 Evolved traceability information model. Links shown as dotted lines are captured in JIRA. Note that all development artifacts are now linked via the *Ticket* and the roadmap has been removed

Trace link granularity Several studies (e.g., [29, 38, 43]) 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

Table 9 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

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 [38]. 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 [28] 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 [52] and visual trace queries [35] also exist.

Introduction of intermediate development artifacts for traceability purposes As described in Sect. 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 organization, 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 [60].

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 [24]. There is research on automation of this process (e.g., [7, 10, 26]) 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 [42]. 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 [34]. Specifically for traceability, the creators of the links are usually not the ones who benefit the most since they already know the system well [43]. 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.

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 POs 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 organization. For example if an organization 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 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 [56] 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 its 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. This is a valid approach to evaluate an artifact such as TracIMo as witnessed in the design science research literature (see, e.g., [50, 51, 64]). There a number of threats to validity applicable to the case study itself which are discussed below using the threats to validity categories defined by Runeson and Höst [58] for case studies.

Construct validity Construct validity aims to verify that the concepts that are researched are understood by subjects of the research. To evaluate the traceability strategy we designed using TracIMo, 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 the traceability strategy designed with TracIMo 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 which where confirmed by the evaluation interviews as well as the collected metrics. 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.

However, it needs to be noted that the benefits of the new traceability strategy to the organization are not the main subject of research in this paper. We used the case study to evaluate the applicability of TracIMo and whether it is possible to measure any benefits. The results of the case study show that this is possible, independent of the concrete long-term benefit of the concrete strategy. The internal validity of the case study as an indicator for TracIMo's applicability is therefore given.

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, e.g., that the company and development team is small and works in an agile manner, might have been conducive to the application of TracIMo. The concrete strategy developed in the case study including the process and traceability goals, the TIM supporting these goals, and the concrete steps in the process are specific to the case company. However, TracIMo itself has not been developed to only fit this company and none of the steps in TracIMo are specific to the organization or to the concrete case we used to evaluate the methodology. We therefore believe that the steps in TracIMo are generic enough and independent of the context. However, further case studies are needed to verify this.

Reliability This validity threat refers to whether the study is repeatable. We have documented our case study process as much as possible. For instance, our interview guide and the detailed description of the case study are available in the supplemental material for this paper [2]. 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 organization.

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 they are applied in practice using a design science approach. We evaluated TracIMo in a case study with an agile development team. This led to the creation of a "ticket-centric" and incremental traceability strategy that was used to effectively create trace links in retrospect. The case study demonstrates that the steps in TracIMo support the design of a traceability strategy that it is applicable in the practical setting of a company.

Our study also shows that the design goals for TracI-Moand how we implemented them support 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. All of these steps are part of TracIMo and were successfully demonstrated in the case study.

The case study revealed several challenges of introducing traceability in practice. We have proposed solutions for these challenges as part of our iterative application of TracIMo in the case study. As part of our future work, we plan to further evaluate TracIMo, particularly in different development contexts, e.g., with larger organizations and with teams using different development approaches. We also believe that the ability of TracIMo to define metrics and measure quantitatively and qualitatively if a traceability strategy is successful is a necessary step towards identifying the return of investment of traceability. We therefore plan to use these future applications of TracIMo to elicit long term benefits of traceability and devise strategies to quantitatively measure the return on investment of traceability based on the set of metrics.

References

1. Amalfitano D, De Simone V, Maietta RR, Scala S, Fasolino AR (2019) Using tool integration for improving traceability management testing processes: an automotive industrial experience. J Softw Evol Process 31(6):e2171
2. Maro S, Steghöfer J-P, Bozzelli P, Muccini H (2021) Supplemental information for "TracIMo: a traceability introduction methodology and its evaluation in an Agile development team". https://doi.org/10.5281/zenodo.4160568
3. Arkley P, Riddle S, Brookes T (2006) Tailoring traceability information to business needs. In: 2006 14th IEEE international requirements engineering conference (RE). IEEE, pp 239–244
4. Asuncion HU, François F, Taylor RN (2007) 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, pp 115–124
5. Asuncion HU, Asuncion AU, Taylor RN (2010) Software traceability with topic modeling. In: Proceedings of the 32nd ACM/IEEE international conference on software engineering, vol. 1. ACM, pp 95–104
6. Biehl JT, Czerwinski M, Czerwinski M, Smith G, Robertson GG (2007) Fastdash: a visual dashboard for fostering awareness in software teams. In: Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, pp 1313–1322
7. Borg M, Runeson P, Ardö A (2014) Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability. Emp Softw Eng 19(6):1565–1616
8. Bouillon E, Mäder P, Philippow I (2013) A survey on usage scenarios for requirements traceability in practice. In: International working conference on requirements engineering: foundation for software quality. Springer, pp 158–173
9. Cleland-Huang J, Hayes JH, Domel JM (2009) Model-based traceability. In: 2009 ICSE workshop on traceability in emerging forms of software engineering. IEEE, pp 6–10
10. Cleland-Huang J, Czauderna A, Gibiec M, Emenecker J (2010) A machine learning approach for tracing regulatory codes to product specific requirements. In: Proceedings of the 32nd

ACM/IEEE international conference on software engineering. ACM, pp 155–164

11. Cleland-Huang J, Gotel OC, Huffman Hayes J, Mäder P, Zisman A (2014) Software traceability: trends and future directions. In: Future of software engineering proceedings, pp 55–69
12. COEST (2015) Center of excellence for software traceability (coest). http://www.coest.org. Accessed 15 Oct 2017
13. Cruzes DS, Dyba T (2011) Recommended steps for thematic synthesis in software engineering. In: 2011 International symposium on empirical software engineering and measurement. IEEE, pp 275–284
14. De Lucia A, Fasano F, Oliveto R (2008) Traceability management for impact analysis. In: Frontiers of software maintenance, 2008. FoSM 2008. IEEE, pp 21–30
15. Dömges R, Pohl K (1998) Adapting traceability environments to project-specific needs. Commun ACM 41(12):54–62
16. Dybå T (2005) An empirical investigation of the key factors for success in software process improvement. IEEE Trans Softw Eng 31(5):410–424
17. Engelsman W, Wieringa RJ, van Sinderen M, Gordijn J, Haaker T (2019) Realizing traceability from the business model to enterprise architecture. In: International conference on conceptual modeling. Springer, pp 37–46
18. Espinoza A, Garbajosa J (2011) A study to support agile methods more effectively through traceability. Innov Syst Softw Eng 7(1):53–69
19. Florez JM (2019) Automated fine-grained requirements-to-code traceability link recovery. In: 2019 IEEE/ACM 41st international conference on software engineering: companion proceedings (ICSE-Companion). IEEE, pp 222–225
20. Gotel O, Mäder P (2012) Acquiring tool support for traceability. In: Software and systems traceability. Springer, pp 43–68
21. Gotel O, Cleland-Huang J, Hayes JH, Zisman A, Egyed A, Grünbacher P, Antoniol G (2012a) The quest for ubiquity: A roadmap for software and systems traceability research. In: 2012 20th IEEE international requirements engineering conference (RE). IEEE, pp 71–80
22. Gotel O, Cleland-Huang J, Hayes JH, Zisman A, Egyed A, Grünbacher P, Dekhtyar A, Antoniol G, Maletic J (2012b) The grand challenge of traceability (v1. 0). In: Software and systems traceability. Springer, pp 343–409
23. Gotel O, Cleland-Huang J, Hayes JH, Zisman A, Egyed A, Grünbacher P, Dekhtyar A, Antoniol G, Maletic J, Mäder P (2012c) Traceability fundamentals. In: Software and systems traceability. Springer, pp 3–22
24. Gotel OC, Finkelstein C (1994) An analysis of the requirements traceability problem. In: 1994., Proceedings of the first international conference on requirements engineering (RE). IEEE, pp 94–101
25. Guo J, Cheng J, Cleland-Huang J (2017) Semantically enhanced software traceability using deep learning techniques. In: Proceedings of the 39th international conference on software engineering. IEEE Press, pp 3–14
26. Hayes JH, Dekhtyar A, Osborne J (2003) Improving requirements tracing via information retrieval. In: 2003 11th IEEE international requirements engineering conference (RE). IEEE, pp 138–147
27. Ingram C, Riddle S (2012) Cost-benefits of traceability. In: Software and systems traceability. Springer, pp 23–42
28. Itemis (2019) Yakindu traceability. https://www.itemis.com/en/yakindu/traceability/. Accessed 07 Aug 2019
29. Javed MA, Zdun U (2014) 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, p 16
30. Jönsson P, Lindvall M (2005) Impact analysis. In: Engineering and managing software requirements. Springer, pp 117–142
31. Kinoshita F (2008) Practices of an agile team. In: Agile 2008 conference. IEEE, pp 373–377
32. Kirova V, Kirby N, Kothari D, Childress G (2008) Effective requirements traceability: models, tools, and practices. Bell Labs Tech J 12(4):143–157
33. Klimpke L, Hildenbrand T (2009) Towards end-to-end traceability: insights and implications from five case studies. In: 2009 Fourth international conference on software engineering advances. IEEE, pp 465–470
34. Kotter JP, Cohen DS (2002) The heart of change: Real-life stories of how people change their organizations. Harvard Business Press, Harvard
35. Mäder P, Cleland-Huang J (2013) A visual language for modeling and executing traceability queries. Softw Syst Modell 12(3):537–553
36. Mäder P, Gotel O (2012) Ready-to-use traceability on evolving projects. In: Software and systems traceability. Springer, pp 173–194
37. Mader P, Gotel O, Philippow I (2009) Motivation matters in the traceability trenches. In: 2009 17th IEEE international requirements engineering conference. IEEE, pp 143–148
38. Mader P, Jones PL, Zhang Y, Cleland-Huang J (2013) Strategic traceability for safety-critical projects. IEEE Softw 30(3):58–66
39. Mahmoud A, Niu N (2013) Supporting requirements traceability through refactoring. In: 2013 21st IEEE international requirements engineering conference (RE). IEEE, pp 32–41
40. Maro S, Steghöfer JP (2016) Capra: a configurable and extendable traceability management tool. In: 2016 24th International requirements engineering conference (RE). IEEE, pp 407–408
41. Maro S, Anjorin A, Wohlrab R, Steghöfer JP (2016) Traceability maintenance: factors and guidelines. In: 2016 31st IEEE/ACM international conference on automated software engineering (ASE). IEEE, pp 414–425
42. Maro S, Steghöfer JP, Hayes J, Cleland-Huang J, Staron M (2018a) Vetting automatically generated trace links: what information is useful to human analysts? In: 2018 IEEE 26th international requirements engineering conference (RE). IEEE, pp 52–63
43. Maro S, Steghöfer JP, Staron M (2018b) Software traceability in the automotive domain: challenges and solutions. J Syst Softw 141:85–110
44. Mezghani M, Kang J, Kang EB, Sedes F (2019) Clustering for traceability managing in system specifications. In: 2019 IEEE 27th international requirements engineering conference (RE). IEEE, pp 257–264
45. Nair S, De La Vara JL, Sen S (2013) A review of traceability research at the requirements engineering conference re@ 21. In: 2013 21st IEEE international requirements engineering conference (RE). IEEE, pp 222–229
46. Nair S, de la Vara JL, Melzi A, Tagliaferri G, De-La-Beaujardiere L, Belmonte F (2014) Safety evidence traceability: Problem analysis and model. In: International working conference on requirements engineering: Foundation for software quality. Springer, pp 309–324
47. Niazi M (2006) Software process improvement: a road to success. In: International conference on product focused software process improvement. Springer, pp 395–401
48. Pages B (2018) Bouml. https://www.bouml.fr/index.html. Accessed 23 May 2019
49. Panis MC (2010) Successful deployment of requirements traceability in a commercial engineering organization... really. In: 2010 18th IEEE InternationalRequirements Engineering Conference (RE), IEEE, pp 303–307

50. Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S (2007) A design science research methodology for information systems research. J Manage Inform Syst 24(3):45–77
51. Prat N, Comyn-Wattiau I, Akoka J (2014) Artifact evaluation in information systems design-science research-a holistic view. In: 9th Pacific Asia conference on information systems. p 23
52. Pruski P, Lohar S, Goss W, Rasin A, Cleland-Huang J (2015) Tiqi: answering unstructured natural language trace queries. Requir Eng 20(3):215–232
53. Ramesh B, Jarke M (2001) Toward reference models for requirements traceability. IEEE Trans Softw Eng 27(1):58–93
54. Regan G, McCaffery F, McDaid K, Flood D (2012) The barriers to traceability and their potential solutions: towards a reference framework. In: 2012 38th Euromicro conference on software engineering and advanced applications. IEEE, pp 319–322
55. Rempel P, Lehnert S, Kuschke T et al (2012) A framework for traceability tool comparison. Softwaretechnik-Trends 32(3):6–11
56. Rempel P, Mäder P, Kuschke T (2013) An empirical study on project-specific traceability strategies. In: 2013 21st IEEE international requirements engineering conference (RE). IEEE, pp 195–204
57. Rempel P, Mäder P, Kuschke T, Cleland-Huang J (2014) Mind the gap: assessing the conformance of software traceability to relevant guidelines. In: Proceedings of the 36th international conference on software engineering. ACM, pp 943–954
58. Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Emp Softw Eng 14(2):131
59. Ståhl D, Hallén K, Bosch J (2017) Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework. Emp Softw Eng 22(3):967–995
60. Staron M (2006) Adopting model driven software development in industry—a case study at two companies. In: Nierstrasz O, Whittle J, Harel D, Reggio G (eds) Model driven engineering languages and systems. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 57–72
61. Steghöfer JP (2017) Software traceability tools: Overview and categorisation. In: Report of the GI working group "traceability/evolution". German Informatics Society (GI), pp 2–7. http://pi.informatik.uni-siegen.de/gi/stt/38_1/01_Fachgruppenberichte/ARC_AKTE/ARC_AKTE_2017_p2_steghoefer.pdf
62. Van Solingen R, Basili V, Caldiera G, Rombach HD (2002) Goal/question/metric (GQM) approach. Encyclopedia of Software Engineering
63. Vara JM, Bollati VA, Jiménez Á, Marcos E (2014) Dealing with traceability in the mddof model transformations. IEEE Trans Softw Eng 40(6):555–583
64. Venable J, Pries-Heje J, Baskerville R (2012) A comprehensive framework for evaluation in design science research. In: International conference on design science research in information systems. Springer, pp 423–438
65. Wang B, Peng R, Li Y, Lai H, Wang Z (2018) Requirements traceability technologies and technology transfer decision support: a systematic review. J Syst Softw 146:59–79
66. Wieringa R (2010) Design science methodology: principles and practice. In: Proceedings of the 32nd ACM/IEEE international conference on software engineering, vol 2. ACM, pp 493–494
67. Winkler S, Pilgrim J (2010) A survey of traceability in requirements engineering and model-driven development. Softw Syst Model (SoSyM) 9(4):529–565
68. Wohlrab R, Steghöfer JP, Knauss E, Maro S, Anjorin A (2016) Collaborative traceability management: challenges and opportunities. In: 2016 IEEE 24th international requirements engineering conference (RE). IEEE, pp 216–225
69. Wolfenstetter T, Basirati MR, Böhm M, Krcmar H (2018) Introducing trails: a tool supporting traceability, integration and visualisation of engineering knowledge for product service systems development. J Syst Softw 144:342–355