Advancing trace recovery evaluation: Applied information retrieval in a software engineering context

Borg, Markus

2012

Advancing Trace Recovery Evaluation – Applied Information Retrieval in a Software Engineering Context

Markus Borg

Licentiate Thesis, 2012
Department of Computer Science
Lund University

Licentiate Thesis 13, 2012
ISSN 1652-4691
Department of Computer Science
Lund University
Box 118
SE-221 00 Lund
Sweden

Email: Markus.Borg@cs.lth.se
WWW: http://cs.lth.se/markus_borg

Cover art: “Exodus from the cave” by Hannah Oredsson
Typeset using LaTeX
Printed in Sweden by Tryckeriet i E-huset, Lund, 2012

ABSTRACT

Successful development of software systems involves the efficient navigation of software artifacts. However, as artifacts are continuously produced and modified, engineers are typically plagued by challenging information landscapes. One state-of-practice approach to structure information is to establish trace links between artifacts; a practice that is also enforced by several development standards. Unfortunately, manually maintaining trace links in an evolving system is a tedious task. To tackle this issue, several researchers have proposed treating the capture and recovery of trace links as an Information Retrieval (IR) problem. The goal of this thesis is to contribute to the evaluation of IR-based trace recovery, both by presenting new empirical results and by suggesting how to increase the strength of evidence in future evaluative studies.

This thesis is based on empirical software engineering research. In a Systematic Literature Review (SLR) we show that a majority of previous evaluations of IR-based trace recovery have been technology-oriented, conducted in "the cave of IR evaluation", using small datasets as experimental input. Also, software artifacts originating from student projects have frequently been used in evaluations. We conducted a survey among traceability researchers and found that while a majority consider student artifacts to be only partly representative of industrial counterparts, such artifacts were typically not validated for industrial representativeness. Our findings call for additional case studies to evaluate IR-based trace recovery within the full complexity of an industrial setting. Thus, we outline future research on IR-based trace recovery in an industrial study on safety-critical impact analysis.

Also, this thesis contributes to the body of empirical evidence of IR-based trace recovery in two experiments with industrial software artifacts. The technology-oriented experiment highlights the clear dependence between datasets and the accuracy of IR-based trace recovery, in line with findings from the SLR. The human-oriented experiment investigates how different quality levels of tool output affect the tracing accuracy of engineers. While the results are not conclusive, there are indications that it is worthwhile further investigating into the actual value of improving tool support for IR-based trace recovery. Finally, we present how tools and methods are evaluated in the general field of IR research, and propose a taxonomy of evaluation contexts tailored for IR-based trace recovery in software engineering.

ACKNOWLEDGEMENTS

This work was funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering.

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Dr. Per Runeson, for all his support and for showing me another continent. I will never forget the hospitality his family extended to me during my winter in North Carolina.

Secondly, thanks go to my assistant supervisor, Prof. Dr. Björn Regnell, for stimulating discussions in the EASE Theme D project. I am also very grateful to Dr. Dietmar Pfahl. Thank you for giving your time to explain all those things I should have known already. I want to thank my co-authors Krzysztof Wnuk, Dr. Anders Ardö, and Dr. Saïd Assar. All of you have shown me how research can be discussed outside the office sphere. Furthermore, I would like to clearly acknowledge my other colleagues at the Department of Computer Science and the Software Engineering Research Group, as well as my colleagues in the EASE Theme D project and in the SWELL research school.

Moreover, as a software engineer on study leave, I would like to recognize my colleagues at ABB in Malmö. In particular, I want to thank my former manager Henrik Holmqvist for suggesting the position as a PhD student. Also, thanks go to both my current manager Christer Gerding for continual support, and to Johan Gren for being my main entry point regarding communication with the real world.

Finally, I want to thank all my family and friends, and to express my honest gratitude to my parents for always helping me with my homework, and without whose support I wouldn't have begun this journey. And I am indebted to Hannah for the cover art and to Ulla and Ronny for wining and dining me while I wrapped up this thesis. And most importantly, thank you to Marie for all the love and being the most stubborn of reviewers. Of course, you have mattered the most!

Markus Borg
Malmö, August 2012

Pursuing that doctoral degree
(V) (;,,;) (V)

LIST OF PUBLICATIONS

Publications included in the thesis

I **Recovering from a decade: A systematic literature review of information retrieval approaches to software traceability**
Markus Borg, Per Runeson, and Anders Ardö
Submitted to a journal, 2012.

II **Industrial comparability of student artifacts in traceability recovery research - An exploratory survey**
Markus Borg, Krzysztof Wnuk, and Dietmar Pfahl
In Proceedings of the 16th European Conference on Software Maintenance and Reengineering (CSMR'12), Szeged, Hungary, pp. 181–190, 2012.

III **Evaluation of traceability recovery in context: A taxonomy for information retrieval tools**
Markus Borg, Per Runeson, and Lina Brodén
In Proceedings of the 16th International Conference on Evaluation & Assessment in Software Engineering (EASE'12), Ciudad Real, Spain, pp. 111–120, 2012.

IV **Do better IR tools improve software engineers' traceability recovery?**
Markus Borg, and Dietmar Pfahl
In Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering (MALETS'11), Lawrence, KS, USA, pp. 27–34, 2011.

The following papers are related, but not included in this thesis. The research agenda presented in Section 8 is partly based on Paper V, which was published as a position paper. Paper VI contains a high-level discussion on information management in a large-scale software engineering context.

Related publications

V **Findability through traceability - A realistic application of candidate trace links?**
Markus Borg
In Proceedings of 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'12), Wrocław, Poland, pp. 173–181, 2012.

VI **Towards scalable information modeling of requirements architectures**
Krzysztof Wnuk, Markus Borg, and Saïd Assar
To appear in the Proceedings of the 1st International Workshop on Modelling for Data-Intensive Computing, (MoDIC'12), Florence, Italy, 2012.

Contribution statement

Markus Borg is the first author of all included papers. He was the main inventor and designer of the studies, and was responsible for running the research processes. Also, he conducted most of the writing.

The SLR reported in Paper I was a prolonged study, which was conducted in parallel to the rest of the work included in this licentiate thesis. The study was co-designed with Prof. Per Runeson, but Markus Borg wrote a clear majority of the paper. Furthermore, Mats Skogholm, a librarian at the Lund University library contributed to the development of search strings, and Dr. Anders Ardö validated the data extraction process and reviewed technical contents of the final report.

The survey in Paper II was conducted by Markus Borg and Krzysztof Wnuk. They co-designed the study and distributed the questionnaire in parallel, however Markus Borg was responsible for the collection and analysis of the data. Markus Borg wrote a majority of the report, however with committed assistance from Dr. Dietmar Pfahl and Krzysztof Wnuk.

The experimental parts of Paper III started as a master thesis project by Lina Brodén. Markus Borg was the main supervisor, responsible for research design and collection of data from industry. Prof. Per Runeson had the role as examiner, and also gave initial input to the study design. The outcome of the master thesis project was then used by Markus Borg as input to Paper III, who extended it with a discussion on evaluation contexts.

The experimental setup in Paper IV was originally designed by Markus Borg, and then further improved together with Dr. Dietmar Pfahl. Markus Borg executed the experiment, analyzed the data, and conducted most of the writing. Dr. Dietmar Pfahl also contributed as an active reviewer and initiated rewarding discussions.

CONTENTS

1 Introduction

Modern society depends on software-intensive systems. Software operates invisibly in everything from kitchen appliances to critical infrastructure, and living a life without daily relying on systems running software requires a determined downshifting from life as most people enjoy it. As the significance of software continuously grows, so does the importance of being able to create it efficiently.

Software development is an inclusive expression used to describe any approach to produce source code and its related documentation. During the software crisis of the 1960s, it became clear that software complexity quickly rises when scaled up to larger systems. The development methods that were applied at the time did not result in required software in a predictable manner. *Software engineering* was coined to denote software developed according to a systematic and organized approach, aiming to effectively produce high-quality software with reduced uncertainty [72]. By applying the engineering paradigm to software development, activities such as analysis, specification, design, implementation, verification, and evolution turned into well-defined practices. On the other hand, additional knowledge-intensive activities tend to increase the number of documents maintained in a project [97].

Large projects risk being characterized by *information overload*, a state where individuals do not have time or capacity to process all available information [33]. Knowledge workers frequently report the stressing feeling of having to deal with too much information [32], and in general spend a substantial effort on locating relevant information [59, 65]. Also, in software engineering the challenge of dealing with a plentitude of software artifacts has been highlighted [37, 74]. Thus, an important characteristic of a software engineering organization is the *findability* it provides, herein defined as "the degree to which a system or environment supports navigation and retrieval" [71], particularly in globally distributed development [28].

One state-of-practice way to support findability in software engineering is to maintain *trace links* between artifacts. *Traceability* is widely recognized as an important factor for efficient software engineering [5, 18, 29]. Traceability supports engineering activities related to software evolution, e.g., change management, impact analyses, and regression testing [5, 16]. Also, traceability assists engineers in less concrete tasks such as system comprehension, knowledge transfer, and process alignment [23, 80, 85]. On the other hand, maintaining trace links in an evolving system is known to be a tedious task [29, 34, 48]. Thus, to support trace link maintenance, several researchers have proposed tool support for *trace recovery*, i.e., proposing candidate trace links among existing artifacts, based on *Information Retrieval* (IR) approaches [5, 23, 48, 68, 70]. The rationale is that IR refers to a set of techniques for finding relevant documents from within large collections [6, 69], and that the search for trace links can be interpreted as an attempt to satisfy an information need.

This thesis includes an aggregation of empirical evidence of *IR-based trace*

recovery, and contributes to the body of knowledge on conducting evaluative studies on IR-based trace recovery tools. As applied researchers, our main interest lies in understanding how feasible trace recovery tools would be for engineers in industrial settings. Paper I contains a comprehensive literature review of previous research on the topic. Based on questions that arose during the literature review, other studies were designed and conducted in parallel. Paper II provides an increased understanding of the validity of using artifacts originating from student projects as experimental input. Paper III reports from an experiment with trace recovery tools on artifacts collected from industry, and also proposes a taxonomy of evaluation contexts tailored for IR-based trace recovery. Finally, Paper IV presents a novel experiment design, addressing the value of slight tool improvements.

This introduction chapter provides a background for the papers and describes relationships between studies. The remainder of this chapter is organized as follows. Section 2 presents a brief background of traceability research and introduces fundamentals of IR. Also, it presents how IR can be applied to address trace recovery. Section 3 expresses the overall aim of this thesis as research questions, and proposes three viewpoints from which the results can be interpreted. Also, Section 4 presents the research methodologies used to answer the research questions. The results of each individual paper are presented in Section 5, while Section 6 draws together the results to provide answers to the research questions. Section 7 highlights some threats to validity in the presented research. In Section 8, we present how we plan to continue our work in future studies. Finally, Section 9 concludes the introduction of this thesis.

2 Background and Related Work

This section presents a background of traceability and IR, the two main research areas on which this thesis rests. Also, we present how IR has been applied to support trace link maintenance. Finally, we present related work on advancing IR-based trace recovery evaluation.

2.1 Traceability - A Fundamental Software Engineering Challenge

The concept of traceability has been discussed in software engineering since the very beginning. Already at the pioneering NATO Working Conference on Software Engineering in 1968, a paper by Randall recognized the need for a developed software system to "contain explicit traces of the design process" [81]. In an early survey of software engineering state-of-the-art in the 1970s by Boehm, "traceability" was mentioned six times, both in relation to engineering challenges at the time, and when predicting future trends [12]. In the software industry, traceability became acknowledged as an important aspect in high quality development.

Consequently, by the 1980s several development standards specified process requirements on the maintenance of traceability information [30]. At the time, the IEEE definition of traceability was "the degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; for example, the degree to which the requirements and design of a given software component match" [50].

In the 1990s, the requirements engineering community emerged and established dedicated publication fora. As traceability was in scope, published research on the topic increased and its relation to requirements engineering was further emphasized. A paper by Gotel and Finkelstein in 1994 identified the lack of a common definition of requirements traceability and suggested: "Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)" [40]. According to a recent systematic literature review by Torkar *et al.*, this definition of requirements traceability is the most commonly cited in research publications [91].

While traceability has been questioned by some of the lean-thinkers of the agile movement in the 2000s to be too costly in relation to its benefits [79], traceability continues to be a fundamental aspect in many development contexts. Since traceability is important to software verification, general safety standards such as IEC 61508 [53], and industry-specific variants (e.g., ISO 26262 in the automotive industry [54] and IEC 61511 in the process industry [52]) mandate maintenance of traceability information. Furthermore, as traceability has a connection to quality, it is also required by organizations aiming at process improvement as defined by CMMI (Capability Maturity Model Integration) [14]. Thus, traceability is neither negotiable in safety certifications nor in CMMI appraisals. In 2006, the international organization CoEST, the Center of Excellence for Software Traceability[1], was founded to gather academics and practitioners to advance traceability research.

In the beginning of 2012, an extensive publication edited by Cleland-Huang *et al.* was published [18], containing contributions from several leading researchers in the traceability community. Apart from summarizing various research topics on traceability, the work presents a number of fundamental definitions. We follow the proposed terminology in the introduction chapter of this thesis. However, when the included Papers II-IV were written, the terminology was not yet aligned in the community. Cleland-Huang *et al.* define traceability as: "the potential for traces to be established and used" [39], i.e., the trace *ability* is stressed. On the other hand, they also present the more specific definition of *requirements traceability* by Gotel and Finkelstein in its original form.

[1] www.coest.org

A number of other terms relevant for this thesis are also defined in the book. A *trace artifact* denotes a traceable unit of data. A *trace link* is an association forged between two trace artifacts, representing relations such as overlap, dependency, contribution, evolution, refinement, or conflict [80]. The main subject in this thesis, *trace recovery*, denotes an approach to create trace links among existing software artifacts. This is equivalent to what we consistently referred to as *traceability recovery* in the Papers II-IV.

2.2 Information Retrieval - Satisfying an Information Need

A central concept in this thesis is *information seeking*, the "conscious effort to acquire information in response to a gap in knowledge" [15]. Particularly, we are interested in finding pieces of information that enable trace links to be recovered, i.e. *trace link seeking*. One approach to seek information is *information retrieval*, meaning "finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)" [69]. This section briefly presents the two main categories of IR models. A longer presentation is available in Paper I.

Typically, IR models apply the bag-of-words model, a simplifying assumption that represents a document as an unordered collection of words, disregarding word order [69]. Most IR models can be classified as either *algebraic* or *probabilistic*, depending on how relevance between queries and documents is measured. Algebraic IR models assume that relevance is correlated with similarity, while probabilistic retrieval is based on models estimating the likelihood of queries being related to documents.

The *Vector Space Model* (VSM), developed in the 1960s, is the most commonly applied algebraic IR model [86]. VSM represents both documents and queries as vectors in a high-dimensional space and similarities are calculated between vectors using distance functions. In principle, every term constitutes a dimension. Usually, terms are weighted using some variant of Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF is used to weight a term based on the length of the document and the frequency of the term, both in the document and in the entire document collection. *Latent Semantic Indexing* (LSI) is an approach to reduce the dimension space, sometimes successful in reducing the effects of synonymy and polysemy [25]. LSI uses singular value decomposition to transform the dimensions from individual terms to combinations of terms, i.e., concepts constitute the dimensions rather than individual terms.

In probabilistic models, documents are ranked according to their probability of being relevant to the query. Two common models are the *Binary Independence retrieval Model* (BIM) [25] [82] and *Probabilistic Inference Networks* [92]. A subset of probabilistic retrieval estimate *Language Models* (LM) for each document. Documents are then ranked based on the probability that a document would generate the terms of a query [78]. A later refinement of simple LMs, topic models,

describes documents as a mixture over topics, where each topic is characterized by an LM. Examples include probabilistic latent semantic indexing [45] and Latent Dirichlet Allocation (LDA) [11].

2.3 Information Retrieval Evaluation

IR is a highly experimental discipline, and empirical evaluations are the main research tool to scientifically compare IR algorithms. The state-of-the-art has advanced through careful examination and interpretation of experimental results. Traditional IR evaluation, as it was developed by Cleverdon in the Cranfield project in the late 1950s [20], consists of three main elements: a document collection, a set of information needs (typically formulated as queries), and relevance judgments telling what documents are relevant to these information needs, i.e., a *gold standard*. Thus, as the experimental units are central in IR evaluation it is important to address threats to content validity, i.e., the extent to which the experimental units reflect and represent the elements of the domain under study [93]. A selection of experimental units that match the real-world setting should carefully be selected, and the sample should be sufficiently large to be representative to the domain.

The most common way to evaluate the effectiveness of an IR system is to measure *precision* and *recall*. As displayed in Figure 1, precision is the fraction of retrieved documents that are relevant, while recall is the fraction of relevant documents that are retrieved. The outcome is often visualized as a Precision-Recall (P-R) curve where the average precision is plotted at fixed recall values, presented as PR@Fix in Figure 1. However, this set-based approach has been criticized for being opaque, as the resulting curve obscure the actual numbers of retrieved documents needed to get beyond low recall [90]. Alternatively, the ranking of retrieval results can be taken into account. The most straightforward approach is to plot the P-R curve for the top k retrieved documents instead [69], shown as PR@k in Figure 1. In such curves, one can see the average accuracy of the first search result, the first ten search results etc. The Text Retrieval Conference (TREC), hosting the most distinguished evaluation series for IR, reports results using both precision at 11 fixed recall values (0.0, 0.1 ... 1.0) and precision at the top 5, 10, 15, 30, 100 and 200 retrieved documents [90]. A discussion on IR-based trace recovery evaluation styles is available in Paper I.

There are several other measures available for IR evaluations, including F-measure, ROC curve, R-precision and the break-even point [69], but none of them are as established as P-R curves. On the other hand, two measures offering single figure effectiveness have gained increased attention. *Mean Average Precision* (MAP), roughly the area under the P-R curve for a set of queries, is established in the TREC community. *Normalized Discounted Cumulative Gain* [56], similar to precision at top k retrieved documents but especially designed for non-binary relevance judgments, is popular especially among researchers employing machine learning techniques to rank search results.

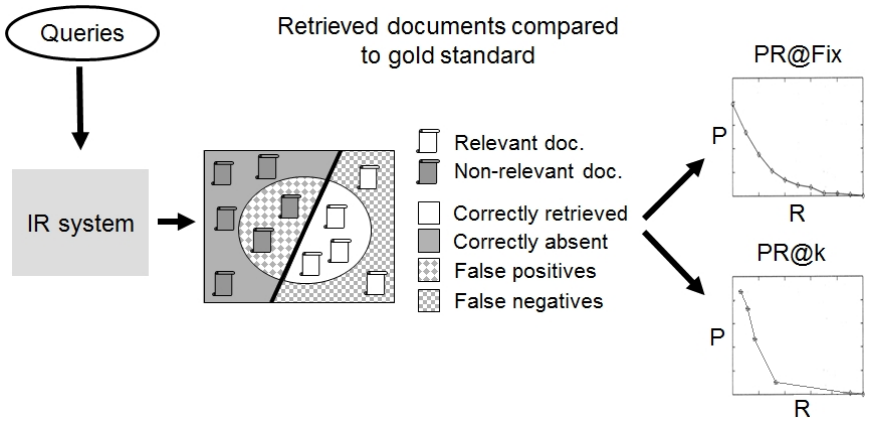


Figure 1: Traditional IR evaluation using P-R curves showing PR@Fix and PR@k. In the center part of the figure, displaying a document space, the relevant items are to the right of the straight line while the retrieved items are within the oval.

The experimental setups of IR evaluations rarely fulfil assumptions required for significance testing, e.g., independence between retrieval results, randomly sampled document collections, and normal distributions. Thus, traditional statistics has not had a large impact on IR evaluation [41]. However, it has been proposed both to use hypergeometric distributions to compute retrieval accuracy to be expected by chance [87], and to apply the Monte Carlo method [13].

Although IR evaluation has been dominated by technology-oriented experiments, it has also been challenged for its unrealistic lack of user involvement [61]. Ingwersen and Järvelin argued that IR is always evaluated in a context and proposed an evaluation framework, where the most simplistic evaluation context is referred to as "the cave of IR evaluation" [51]. In Paper III, we present their framework and an adapted version tailored for IR-based trace recovery.

2.4 Trace Recovery - An Information Retrieval Problem

Tool support for linking artifacts containing Natural Language (NL) has been explored by researchers since at least the early 1990s. Whilst a longer history of IR-based trace recovery is available in Paper I, this section introduces the approach and highlights some key publications.

The underlying assumption of using IR for trace recovery is that artifacts with highly similar textual content are good candidates to be linked. Figure 2 shows the key steps involved in IR-based trace recovery, organized in a pipeline architecture as suggested by De Lucia *et al.* [24]. First, input documents are parsed and preprocessed, typically using *stop word removal* and *stemming*. In the second step, the

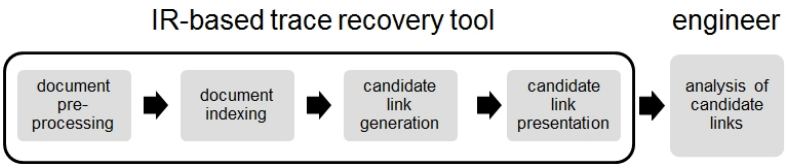


Figure 2: Key steps in IR-based trace recovery, adapted from De Lucia *et al.* [24].

documents are indexed using the IR model. Then, candidate trace links are generated, ranked according to the IR model, and the result is visualized. Finally, the result is presented to the engineer, as emphasized in Figure 2, who gets to assess the output. The rationale is that it is faster for an engineer to assess a ranked list of candidate trace links, despite both missed links and false positives, than seeking trace links from scratch.

In 1998, a pioneering study by Fiutem and Antoniol proposed a trace recovery process to bridge the gap between design and code, based on edit distances between NL content of artifacts [36]. They coined the term "traceability recovery", and published several papers on the topic. Also, they were the first to express identification of trace links as an IR problem [4]. Their well-cited work from 2002 compared the accuracy of candidate trace links from two IR models, BIM and VSM [5]. Marcus and Maletic were the first to apply LSI to recover trace links between source code and NL documentation [70]. Huffman Hayes *et al.* enhanced trace recovery based on VSM with relevance feedback. They had from early on a human-oriented perspective, aiming at supporting V&V activities at NASA using their tool RETRO [49]. De Lucia *et al.* have conducted work focused on empirically evaluating LSI-based traceability recovery in their document management system ADAMS [22]. They have advanced the empirical foundation by conducting a series of controlled experiments and case studies with student subjects. Cleland-Huang and colleagues have published several studies using probabilistic IR models for trace recovery, implemented in their tool Poirot [66]. Much of their work has focused on improving the accuracy of their tool by various enhancements.

Lately, a number of publications suggest that the P-R differences for trace recovery between different IR models are minor. Oliveto *et al.* compared VSM, LSI, LM and LDA on artifacts originating from two student projects, and found no significant differences [73]. Also a review by Binkley and Lawrie reported the same phenomenon [10]. Falessi *et al.* proposed a taxonomy of algebraic IR models and experimentally studied how differently configured algebraic IR models performed in detecting duplicated requirements [34]. They concluded that simple IR solutions tend to produce more accurate output.

We have identified some progress related to IR-based trace recovery in non-academic environments. In May 2012, a US patent with the title "System and method for maintaining requirements traceability" was granted [9]. The patent ap-

plication, filed in 2007, describes an application used to synchronize artifacts in a requirements repository and a testing repository. Though the actual linking process is described in general terms, a research publication implementing trace recovery based on LSI is cited. Also indicating industrial interest in IR-based trace recovery is that HP Quality Center, a component of HP Application Lifecycle Management, provides a feature to link artifacts based on textual similarity analysis [44]. While it is implemented to detect duplicate defect reports, the same technique could be applied for trace recovery.

2.5 Previous Work on Advancing IR-based Trace Recovery Evaluation

While there are several general guidelines on software engineering research (e.g., experiments [95], case studies [84], reporting [57], replications [88], literature reviews [62]), only few publications have specifically addressed research on IR-based trace recovery. The closest related work is described in this section.

Huffman Hayes and Dekhtyar proposed a framework for comparing experiments on requirements tracing [47]. Their framework describes the four experimental phases: definition, planning, realization and interpretation. We evaluated the framework in Paper III, and concluded that it provides valuable structure for conducting technology-oriented experiments. However, concerning human-oriented experiments, there is room for enhancements. Huffman Hayes *et al.* also suggested categorizing trace recovery research as either *studies of methods* (are the tools capable of providing accurate results fast?) or *studies of human analysts* (how do humans use the tool output?) [48]. Furthermore, in the same publication, they suggested assessing the accuracy of tool output according to the quality intervals "Acceptable/Good/Excellent", with specified P-R levels. In Paper I, we catalog the primary publications according to their suggestions, but we also catalog the primary publications according to the context taxonomy we propose in Paper III. A recent publication by Falessi *et al.* proposed seven "empirical principles" for technology-oriented evaluation of IR tools in software engineering, explicitly mentioning trace recovery as one application [35]. Despite the absence of statistical analysis in traditional IR evaluation [41], they argued for both increased difference and equivalence testing. In Paper IV, we also propose equivalence testing of IR-based trace recovery, however in the context of human-oriented experiments.

A number of researchers connected to CoEST have repeatedly argued that a repository of benchmarks for trace recovery research should be established, in line with what has driven large scale IR evaluations at TREC [17,26,27]. Furthermore, Ben Charrada *et al.* have presented a possible benchmark for traceability studies, originating from an example in a textbook on software design [8]. We support the attempt to develop large public datasets, but there are also risks involved in benchmarks. As mentioned in Section 2.3, the results of IR evaluations depend on the experimental input. Thus, there is a risk of over-engineering tools on datasets that

Tool developer (Paper I, Paper IV)	**Which IR model should we implement in our new trace recovery tool?**
RQ1	Which IR model has most frequently been implemented in research tools?
RQ2	Which IR model has displayed the most promising results?
Development manager (Paper I, II)	**Should we deploy an IR-based trace recovery tool in our organization?**
RQ3	What evidence is there that IR-based trace recovery is feasible in an industrial setting?
Traceability researcher (Paper I, III)	**How can we strengthen the base of empirical evidence of IR-based trace recovery?**
RQ4	How can we advance technology-oriented studies on IR-based trace recovery?
RQ5	How can we advance human-oriented studies on IR-based trace recovery?

Table 1: Viewpoints further broken down into RQs. The scope of trace recovery is implicit in RQ1-2, while explicit in RQ3-5.

do not have acceptable content validity. Benchmarking in IR-based trace recovery is further discussed in Paper III.

Another project, also promoted by CoEST, is the TraceLab project [17, 60]. TraceLab is a visual experimental workbench, highly customized for traceability research. It is currently under Alpha release to project collaborators. CoEST claims that it can be used for designing, constructing, and executing trace recovery experiments, and facilitating evaluation of different techniques. TraceLab is said to be similar to existing data mining tools such as Weka and RapidMiner, and aims at providing analogous infrastructure to simplify experimentation on traceability. However, to what extent traceability researchers are interested in a common experimental workbench remains an open question.

3 Research Focus

This section describes how this thesis contributes to the body of knowledge on IR-based trace recovery in general, and evaluations of IR-based trace recovery in particular. We base the discussion on the viewpoints from the perspectives of three stakeholders, each with his own pictured consideration: (1) a CASE tool developer responsible for a new trace recovery tool, (2) a manager responsible for a large software development organization, and (3) an academic researcher trying to advance the trace recovery research frontier. We further divide each viewpoint into more specific Research Questions (RQ), as presented in Table 1.

As the included papers are closely related, all papers to some extent contribute to the three viewpoints. Paper I is the most recent publication included in this thesis, and by far the most comprehensive. As such, it contributes to all individual RQs presented in Table 1. Papers II, III, and IV primarily address the RQs from the perspective of a development manager, an academic researcher, and a tool developer respectively. Figure 3 positions this thesis in relation to existing research on IR and its application on traceability. The rows in the figure show three different research foci: development and application of retrieval models, improving technology-oriented IR evaluations, and improving human-oriented IR evaluations. The arrows A-C indicate how approaches from the IR domain have been applied in traceability research.

- The A arrow represents pioneering work on applying IR models to trace recovery, such as presented by Antoniol *et al.* [5] and Marcus and Maletic [70].
- The B arrow denotes contributions to technology-oriented evaluations of IR-based trace recovery, inspired by methods used in the general IR domain. Examples include the experimental framework proposed by Huffman Hayes and Dekhtyar [47] and CoEST's ambition to create benchmarks [17].
- The context taxonomy we propose in Paper III is an example of work along arrow C, where we apply results from human-oriented IR evaluations to traceability research.

To the right in Figure 3, the internal relations among the included papers are presented. The study in Paper II on industrial comparability of student artifacts was initiated to further explore early results from the work in Paper I. The experiment on human subjects in Paper IV was designed to further analyze the differences between the technology-oriented experimental results in Paper III. Finally, the evaluation taxonomy proposed in paper IV was used in Paper I to structure parts of the literature review.

The right box in Figure 3 also shows to which research focus the included papers mainly contribute. Paper I contributes to all foci. Paper II mainly contributes to improving technology-oriented IR evaluations, as it addresses the content validity of experimental input originating from student projects. Paper III proposes a taxonomy of evaluation contexts, and thus contributes to improving human-oriented IR evaluations. Finally, while Paper IV questions the implications of minor differences of tool output in previous technology-oriented experiments, we consider it to mainly contribute by providing empirical results on the application of IR models on a realistic work task involving trace recovery.

4 Method

The research in this thesis is mainly based on *empirical research*, a way to obtain knowledge through observing and measuring phenomena. Empirical studies result

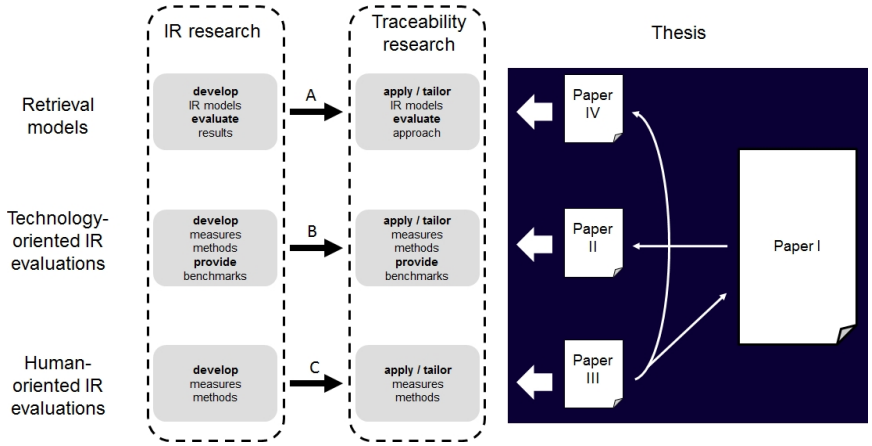


Figure 3: Contributions of this thesis, presented in relation to research on IR and traceability. Arrows A-C denote examples of knowledge transfer between the domains. The right side of the figure positions the individual papers of this thesis, and shows their internal relationships.

in evidence, pieces of information that support conclusions. Since evidence is needed to build and verify theories [63], empirical studies should be conducted to improve the body of software engineering knowledge [31, 84, 89]. However, as engineers, we also have an ambition to create innovations that advance the field of software engineering. One research paradigm that seeks to create innovative artifacts to improve the state-of-practice is *design science* [43]. Design science originates from engineering and is fundamentally a problem solving paradigm. As presented in Figure 4, the build-evaluate loop is central in design science (in some disciplines referred to as action research [83]). Based on empirical understanding of a context, innovations in the form of tools and practices are developed. The innovations are then evaluated empirically in the target context. These steps might then be iterated until satisfactory results have been reached.

This thesis mainly contains *exploratory* and *evaluative* empirical research, based on studies using *experiments*, *surveys*, and *systematic literature reviews* as research methodology. However, also *case studies* are relevant for this thesis, as such studies are discussed as possible future work in Section 8. Future work also involves *design tasks* to improve industry practice, after proper assessment.

Exploratory research is typically conducted in early stages of research projects, and attempts to bring initial understanding to a phenomenon, preferably from rich qualitative data [31]. An exploratory study is seldom used to draw definitive conclusions, but is rather used to guide further work. Decisions on future study design, data collection methods, and sample selections can be supported by preceding exploratory research. Paper I and Paper II explored both tool support for IR-based

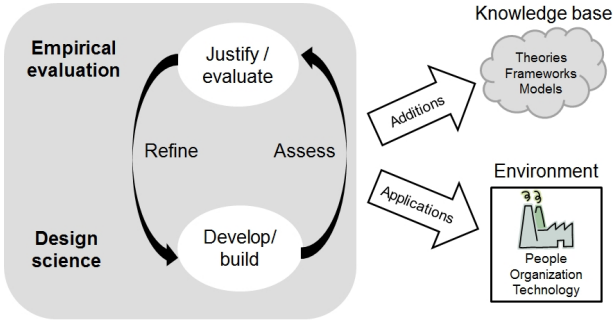


Figure 4: Design science and the build-evaluate loop. Adapted from Hevner *et al.* [43]

trace recovery in general, and evaluation of such tools in particular.

Evaluative research can be conducted to assess the effects and effectiveness of innovations, interventions, practices etc. [83]. An evaluative study involves a systematic collection of data, which can be of both qualitative and quantitative nature. Paper III contains a quantitative evaluation of two IR-based trace recovery tools compared to a naïve benchmark. Paper IV reports from an evaluation comparing how human subjects solve a realistic work task when supported by IR-based trace recovery. Both papers use industrial artifacts as input to the evaluative studies.

4.1 Research Methods

Experiments (or controlled experiments) are commonly used in software engineering to investigate the cause-effect relationships of introducing new methods, techniques or tools. Different treatments are applied to or by different subjects, while other variables are kept constant, and the effects on outcome variables are measured [95]. Experiments are categorized as either *technology-oriented* or *human-oriented*, depending on whether objects or human subjects are given various treatments. Involving human subjects is expensive, consequently university students are commonly used and not engineers working in industry [46]. Paper III presents a technology-oriented experiment, while Paper IV describes an experimental setup of a human-oriented experiment, and results from a pilot run using both students and senior researchers as subjects.

A *case study* in software engineering is conducted to study a phenomenon within its real-life context. Such a study draws on multiple sources of evidence to investigate one or more instances of the phenomenon, and is especially applicable when the boundary between phenomenon and its context cannot be clearly specified [84]. While this thesis does not include any case studies, such studies are planned as future work, and are further discussed in Section 8.

Works	Type of research	Research method
Paper I	Exploratory	Systematic literature review
Paper II	Exploratory	Questionnaire-based survey
Paper III	Evaluative	Technology-oriented experiment
Paper IV	Evaluative	Human-oriented experiment

Table 2: Type and method of research in the included papers.

A *Systematic Literature Review* (SLR) is a secondary study aimed at aggregating a base of empirical evidence. It is an increasingly popular method in software engineering research, relying on a rigid search and analysis strategy to ensure a comprehensive collection of publications [62]. A variant of an SLR is a Systematic Mapping (SM) study, a less granular literature study, designed to identify research gaps and direct future research [62, 76]. The primary contribution of this thesis comes from the SLR presented in Paper I.

Survey research is used to describe what exists, and to what extent, in a given population [55]. Three distinctive characteristics of survey research can be identified [77]. First, it is used to quantitatively (sometimes also qualitatively) describe aspects of a given population. Second, the collected data is collected from people and thus subjective. Third, survey research uses findings from a portion of a population to generalize back to the entire population. Surveys can be divided into two categories based on how they are executed: written surveys (i.e., questionnaires) and verbal surveys (i.e., telephone or face-to-face interviews). Paper II reports how valid traceability researchers consider studies on student artifacts to be, based on empirical data collected using a questionnaire-based survey.

Table 2 summarizes the type of research, and the selected research method, in the included papers.

5 Results

This section presents the main results from each of the included papers.

Paper I: A Systematic Literature Review of IR-based Trace Recovery

The objective of the study was to conduct a comprehensive review of IR-based trace recovery. Particularly focusing on previous evaluations, we explored collected evidence of the feasibility of deploying an IR-based trace recovery tool in an industrial setting. Using a rigorous methodology, we aggregated empirical data from 132 studies reported in 79 publications. We found that a majority of the publications implemented algebraic IR models, most often the classic VSM. Also,

we found that no IR model regularly outperforms VSM. Most studies do not analyze the usefulness of the IR-based trace recovery further than tool output, i.e., evaluations conducted "in the cave", entirely based on P-R curves dominate. The strongest evidence of the benefits of IR-based trace recovery comes from a number of controlled experiments. Several experiments report that subjects perform certain software engineering work tasks faster (and/or with higher quality) when supported by IR-based trace recovery tools. However, the experimental settings have been artificial using mainly student subjects and small sets of software artifacts. In technology-oriented evaluations, we found a clear dependence between datasets used in evaluations and the experimental results. Also, we found that few case studies have been conducted, and only one in an industrial context. Finally, we conclude that the overall quality of reporting should be improved regarding both context and tool details, measures reported, and use of IR terminology.

Paper II: Researchers' Perspectives on the Validity of Student Artifacts

While conducting the SLR in Paper I, we found that in roughly half of the evaluative studies on IR-based trace recovery, output from student projects was used as experimental input. Paper II explores to what extent student artifacts differ from industrial counterparts when used in evaluations of IR-based trace recovery. In a survey among authors identified in the SLR in Paper I, including both academics and practitioners, we found that a majority of the respondents consider software artifacts originating from student projects to be only partly comparable to industrial artifacts. Moreover, only few respondents reported that they validated student artifacts for industrial representativeness before using them as experimental input. Also, our respondents made suggestions for improving the description of artifact sets used in IR-based trace recovery studies.

Paper III: A Taxonomy of Evaluation Contexts and a Cave Study

Paper III contains a technology-oriented experiment, an evaluation "in the cave", of two publicly available IR-based trace recovery tools. We use both a de-facto traceability benchmark originating from a NASA project, and artifacts collected from a company in the domain of process automation. Our study shows that even though both artifact sets contain software artifacts from embedded development, their characteristics differ considerably, and consequently the accuracy of the recovered trace links. Furthermore, Paper III proposes a context taxonomy for evaluations of IR-based trace recovery, covering evaluation contexts from "the cave" to in-vivo evaluations in industrial projects. This taxonomy was then used to structure parts of the SLR in Paper I.

Paper IV: Towards Understanding Minor Tool Improvements

Since a majority of previous studies evaluated IR-based trace recovery only based on P-R curves, one might wonder to what extent minor improvements of tool output actually influence engineers working with the tools. Is it worthwhile to keep hunting slight improvements in precision and recall? To tackle this question, we conducted a pilot experiment with eight subjects, supported by tool output from the two tools evaluated in Paper IV. As such, the subjects were supported by tool output matching two different P-R curves. Inspired by research in medicine, more specifically a study on vaccination coverage [7], we then analyzed the data using statistical testing of equivalence [94]. The low number of subjects did not result in any statistically significant results, but we found that the effect size of being supported by the slightly more accurate tool output was of practical significance. While our results are not conclusive, the pilot experiment indicates that it is worthwhile to investigate further into the actual value of improving tool support for trace recovery, in a replication with more subjects.

6 Synthesis

This section presents a synthesis of the results from the included papers to provide answers to the research questions asked in this thesis.

RQ1: Which IR model has most frequently been implemented in research tools?

Paper I concludes that algebraic IR models have been implemented more often than probabilistic IR models. Although there has been an increasing trend of trace recovery based on probabilistic LMs the last five years, a majority of publications report IR-based trace recovery using vector space retrieval. In roughly half of the papers applying VSM, the number of dimensions of the vector space is reduced using LSI. However, it is important to note that one reason for the many studies on vector space retrieval is that it is frequently used as a benchmark when comparing the output from more advanced IR models.

RQ2: Which IR model has displayed the most promising results?

As presented in Paper I, no IR model has been reported to repeatedly outperform the classic VSM developed in the 60s. This confirms previous work by Oliveto *et al.* [73], Binkley and Lawrie [10], and Falessi *et al.* [34]. Instead, our work shows that the input software artifacts have a much larger impact on the outcome of trace recovery experiments than the choice of IR model. While this is well known in

IR research [69], and has been mentioned by Ali *et al.* in the traceability community [2], it has not been highlighted as clearly before in traceability research.

RQ3: What evidence is there that IR-based trace recovery is feasible in an industrial setting?

The software engineering literature identified in Paper I does not contain any substantial success stories from in-vivo evaluations. Only one industrial case study, conducted in a short 5-people project, has reported that IR-based trace recovery was beneficial. Apart from this study, the strongest empirical evidence comes from controlled experiments with student subjects (similar to our contribution in Paper IV) and case studies in student projects. While these studies suggest that certain traceability-centric work tasks can be supported by IR-based trace recovery tools, the majority of studies do not go further than reporting P-R curves "in the cave of IR evaluation" (similar to our contribution in Paper III). However, some identified non-academic activity indicates a usefulness of the approach. In May 2012, a patent was granted protecting a "System and method for maintaining requirements traceability" [9]. Furthermore, the CASE tool HP Quality Center describes an IR feature in its marketing of the product [44].

RQ4: How can we advance technology-oriented studies on IR-based trace recovery?

As the results of IR-based trace recovery are so dependent on the input software artifacts, there is little value in additional evaluations based on a small number of artifacts. It is critical to conduct experiments on large, preferably publicly available, datasets. While this has been proposed by members of COEST before [8, 17, 26, 27], Paper I underlines how few previous evaluations have been conducted using datasets of reasonable size. Moreover, in Paper II we argue that if student artifacts are to be used as experimental input, they should first be properly validated for industrial representability. In the IR sub-domain of enterprise search, it has been proposed to extract documents from companies that no longer exist [42] (e.g., Enron), an option that could be explored also in software engineering. In Paper I we argue that the reporting of technical details of IR implementations should be improved, while Paper II stresses the importance to clearly describe the input artifacts in technology-oriented experiments. Describing the artifacts is especially important in studies where the artifacts cannot be disclosed, e.g., for confidentiality reasons, as it obstructs secondary studies.

Viewpoint	Consideration	Recommendation
Tool developer	Which IR model should we implement in our new trace recovery tool?	The classic VSM, since several efficient implementations are available as open source. There is no empirical evidence that more advanced IR models produce more accurate trace links.
Development manager	Should we deploy an IR-based trace recovery tool in our organization?	Await empirical evidence from future in-vivo studies. In the meantime, assure that your general search solutions make trace artifacts findable.
Traceability researcher	How can we strengthen the base of empirical evidence of IR-based trace recovery?	Case studies in industrial settings are required. Furthermore, larger datasets containing industrial artifacts should be used as experimental input. Also, the reporting of evaluation contexts, input artifacts, and IR solutions should be improved.

Table 3: Recommendations for the proposed viewpoints.

RQ5: How can we advance human-oriented studies on IR-based trace recovery?

As paper I shows, a majority of evaluations of IR-based trace recovery have been conducted in "the cave of IR evaluation", drawing conclusions based on P-R curves. More evaluations with human subjects, working with realistic tasks, are needed to strengthen the evidence of IR-based trace recovery. While a number of controlled experiments have been conducted, conspicuously few industrial case studies have been reported. In an attempt to guide future studies beyond "the cave", Paper III proposes a taxonomy of evaluation contexts, along with suggested measures, tailored for IR-based trace recovery, based on previous work by Ingwersen and Järvelin [51].

In Table 3, we further summarize our answers in an attempt to address the three viewpoints presented in Section 3. The last column presents our recommendations, based on the understanding obtained during the work of this thesis.

7 Threats to Validity

The results of any research effort should be questioned, even though proper research methodologies were applied. The validity of the research is the foundation on which the trustworthiness of the results is established. In this thesis, threats to validity, and actions taken to reduce the threats, are discussed based on the classification proposed by Wohlin *et al.* [95]. Further details on validity threats are available in the individual papers.

Construct validity is concerned with the relation between theories behind the research and the observations. Consequently, it covers the choice and collection of measures for the studied concepts. For example, the questions of a questionnaire must not be misunderstood or misinterpreted by the respondents. One strategy to increase construct validity is to use multiple sources of evidence, and to establish chains of evidence [96].

In Paper I, we partly aggregate evidence from previous evaluations based on data in tables or directly from P-R curves. Thus, we were limited by the levels of detail included in the reviewed publications. A possible way to obtain richer data would have been to contact the corresponding authors and ask for access to all measurements from the studies. As 79 publications from the last decade were included, it would have required a large effort. On the other hand, as we extracted data from P-R values from 48 publications and, whenever possible, followed the TREC convention of reporting both precision at fixed recall levels as well as precision and recall at certain cut off levels, we limit this threat. Regarding the survey in Paper II, the questionnaire was reviewed by a native English speaker, and a pilot study was conducted on five senior software engineering researchers.

Internal validity is related to issues that may affect the causal relationship between treatment and outcome. In experiments, used in both Paper III and IV, the internal validity questions whether the effect is caused by the independent variables or other factors. Internal validity is typically not a threat to descriptive or exploratory studies, as casual claims rarely are made [96].

The SLR in Paper I is subject to a number of threats to internal validity. As most evaluations of IR-based trace recovery have been conducted in controlled settings, e.g., in university classrooms, we have not considered different domains in the analysis of the results. Further research is required to study whether the approach is more feasible in certain contexts such as safety-critical development. Also, as the use of terminology in the publications was not aligned, our choice of search string might have influenced the resulting evidence base. These threats were addressed by combining database searches with snowball sampling, and by incrementally developing the search string based on a gold standard of publications. Another threat to the SLR is publication bias, e.g., authors might be less likely to publish negative results, or IR-based trace recovery might be successfully used in industry even though it is not reported in research publications. As the results in Paper I are related to all RQs, so are the threats to internal validity. Furthermore,

regarding the experiments included in this thesis, our understanding of the studied IR-based trace recovery tools (Paper III), and the subjects' understanding of the work task (Paper IV), are confounding factors. In both experiments we addressed threats to internal validity by running pilot experiments.

External validity concerns the ability to generalize the findings outside the actual scope of the study. Results obtained in a specific context may not be valid in other contexts. Strategies to address threats to external validity include studying multiple cases and replicating experiments [96].

In Paper II, we surveyed published researchers in the traceability community. However, as the number of respondents was low, we do not have a strong basis for generalizing our results to the entire population of traceability researchers. On the other hand, considering the exploratory nature of our study, the external validity of the survey is acceptable. Another threat to external validity is that all software artifacts used as experimental input in Paper III and IV originate from embedded development contexts, either from the space domain or process automation. Furthermore, as emphasized in Paper I, the limited number of artifacts makes generalizations to larger document spaces uncertain.

Conclusion validity results from the ability to draw correct conclusions about the relation between the treatment and the outcome. This type of validity is related to the repeatability of a study. Threats to conclusion validity in quantitative studies are often related to statistics [95]. In qualitative studies on the other hand, it can be used to discuss to which extent the data and the analysis are dependent on the specific researchers, then sometimes also referred to as reliability [84, 96].

In Paper III, the technology-oriented experimental results were not analyzed using significance testing since assumptions underlying statistical treatment such as independence, random sampling and normality were not met. Instead, the output differences were analyzed in a human-oriented experiment in Paper IV. While these results were analyzed using statistical testing, the low number of subjects did not result in any statistically significant results. On the other hand, we consider the effect sizes reported in Paper IV to be of practical significance.

In Paper I, we assess the strength of evidence of the industrial feasibility of IR-based trace recovery. This assessment involves interpretation. While we do not consider P-R curves from small evaluations "in the cave" to be particularly strong pieces of evidence, other researchers might value them differently. For example, the foreword by Finkelstein in the recently published textbook on software and systems traceability discusses IR-based trace recovery in a less critical manner [18]. However, in line with practices in reflexive methodology, there is a demand for reflection in research in conjunction with interpretation [3]. As such, a researcher should be aware of, and critically confront, favored lines of interpretation. Naturally, there is a risk of bias in the foreword of a textbook, written by a notable part of the traceability research community. In such a foreword, there are few incentives to present sceptical views on the research. On the other hand, there is also a risk that the work in this licentiate thesis, written by a junior PhD student fostered

in a strictly empirical research tradition, is overly critical to the mainly technology-oriented research strategy. Our conclusion, that there is a need for evaluations that go beyond "the cave", might be in the interest of the individual researcher, as an attempt to pave the way for future empirical studies. To conclude, the conclusion validity is a threat to RQ5 and RQ6, as other researchers might suggest different ways to advance evaluation of IR-based trace recovery.

8 Agenda for Future Research

This section presents a speculative research agenda for future work, partly based on Paper V. We intend to continue our research with a focus on trace links, however in a more solution-oriented manner. Our ambition is to study a specific work task that requires an engineer to explicitly specify trace links among artifacts, namely change impact analysis in a safety-critical context. As we suspect that software engineers are more comfortable navigating the source code than its related documentation, we intend to focus specifically on trace links between non-code artifacts. A summary of the planned work in this section is presented as Future research Questions (FQ) and planned Design science Tasks (DT) in Table 5.

8.1 Description of the Context

The targeted impact analysis process is applied by a large multinational company active in the power and automation sector. The development context is safety-critical embedded development in the domain of industrial control systems, governed by IEC 61511 [52]. The number of developers is in the magnitude of hundreds; a project has typically a length of 12-18 months and follows an iterative stage-gate project management model. Also, the software is certified to a Safety Integrity Level (SIL) of 2 as defined by IEC 61508 [53], corresponding to a risk reduction factor of 1.000.000-10.000.000 for continuous operation. Process requirements mandate maintenance of traceability information, especially between requirements and test cases. Both requirements and test case descriptions are predominantly specified in English NL text.

As specified in IEC 61511 [52], impact of proposed software changes, e.g., for error corrections, should be analyzed before implementation. In the initially studied case, as presented in Paper V, this process is integrated in the issue tracking system. As part of the analysis, engineers are required to investigate impact, and report their results according to a project specific template, validated by an external certifying agency. A slightly modified version of this template, recently described as part of a master thesis project [64], is presented in Table 4. As seen in Table 4, several questions explicitly ask for trace links (6 out of 13 questions). The engineer is required to specify source code that will be modified (with a file-level granularity), and also which related software artifacts need to be updated to

Impact Analysis Questions for Error Corrections	
1)	Is the reported problem safety critical?
2)	In which versions/revisions does this problem exist?
3)	How are general system functions and properties affected by the change?
4)	**List modified code files/modules and their SIL classifications, and/or affected safety safety related hardware modules.**
5)	How are general system functions and properties affected by the change?
6)	**Which library items are affected by the change? (e.g., library types, firmware functions, HW types, HW libraries)**
7)	**Which documents need to be modified? (e.g., product requirements specifications, architecture, functional requirements specifications, design descriptions, schematics, functional test descriptions, design test descriptions)**
8)	**Which test cases need to be executed? (e.g., design tests, functional tests, sequence tests, environmental/EMC tests, FPGA simulations)**
9)	**Which user documents, including online help, need to be modified?**
10)	How long will it take to correct the problem, and verify the correction?
11)	What is the root cause of this problem?
12)	How could this problem been avoided?
13)	**Which requirements and functions need to be retested by product test/system test organization?**

Table 4: Impact analysis template. Questions in bold fonts require explicit trace links to other artifacts. Based on a description by Klevin [64].

reflect the changes, e.g., requirement specifications, design documentation, test case descriptions, test scripts and user manuals. Furthermore, the impact analysis should specify which high-level system requirements cover the involved features, and which test cases should be executed to verify that the changes are correct once implemented in the system. Consequently, the impact analysis reports explicitly connect requirements and test artifacts. As this has been reported as a specific challenge in requirements and verification alignment [85], we also intend to explore how the knowledge embedded in the impact analysis reports can be used to support this aspect of large-scale software development.

8.2 Solution idea

While an important part of the impact analysis work task involves specifying trace links to related software artifacts, there are rarely any traceability matrices to consult. Consequently, if engineers do not already know which artifacts are impacted,

a substantial part of the impact analysis work task turns into an information seeking activity. In Figure 5, we present an initial model of the trace link seeking activity involved in the impact analysis. At first, depicted in the left of the figure, the engineer starts the work task with six questions that require explicit trace links. The engineer then enters the process of trace link seeking, presented as the second step in Figure 5. Typically, this is an iterative process where the engineer seeks information suggesting trace links in different ways. Knowledge embedded in previous impact analysis reports can be reused, project documentation can be studied, and colleagues can be asked. As reported by Dagenais *et al.*, especially junior engineers and newcomers rely on communication with more experienced colleagues, in particular when project findability is low due to poor search solutions [21]. Finally, as presented to the right in Figure 5, enough information has been found to specify required trace links in the impact analysis template. As presented in Table 5, we intend to improve the trace link seeking model (DT1) based on observational studies with protocol analysis. This work could complement Freund *et al.*'s more general work on modeling the information behavior of software engineer [37] by exploring a specific work task. Moreover, we plan to assess whether the trace link seeking model is applicable to other contexts with strict process requirements on maintenance of traceability information (FQ1).

Currently, as presented in Paper V, engineers conduct the trace link seeking supported by a low level of automation [75]. Our plan is to increase the level of automation in two areas of the trace link seeking process, as indicated by the cogwheels in Figure 5. In the present work flow, engineers use the search features (primarily keyword-based) of the issue tracking system and the document management system to gather enough information to specify trace links. Our hypothesis is that these steps could be supported by a recommendation system based on textual similarity analysis. As discussed in Paper V, our goal is to support trace link seeking by deploying a plug-in to the issue tracking system (presented as DT2 in Table 5). Developing plug-ins to tools already deployed in industry enables in-vivo studies without introducing additional external tools.

Another direction we want to explore is to consider artifact meta-information to improve the trace recovery, presented as FQ2 in Table 5. One possibility, that we initially have explored, is to exploit the already existing link structures among software artifacts. Using link mining, we have explored clusters of issue reports from the public Android issue tracking system. Figure 6 visualizes link structures among Android issue reports, extracted from hyperlinks manually established by developers. We expect to find patterns of linked artifacts also in the targeted safety-critical case, however also between different types of artifacts, when conducting link mining in the impact analysis reports in the issue tracking system. As hyperlinks have proven useful in tasks such as object ranking, link prediction, and subgraph discovery [38], we hope it can also be used to advance trace recovery. A link mining approach might move our research closer to work on semantic networks of software artifacts, which previously has been used to significantly improve search-

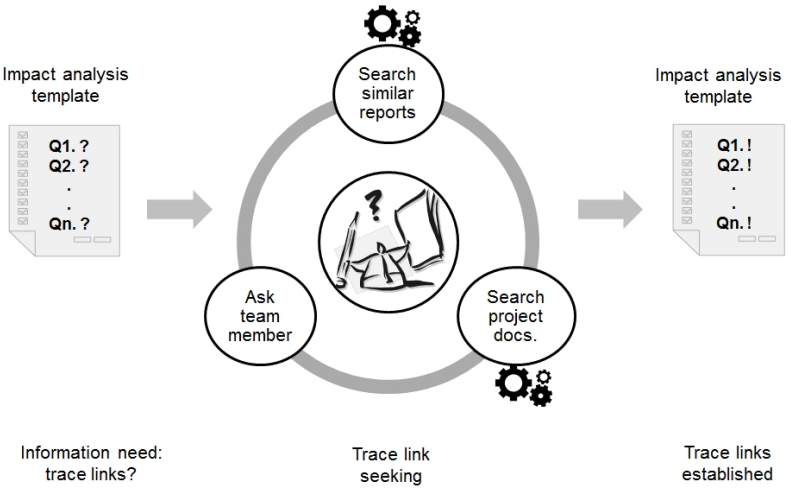


Figure 5: Trace link seeking in the impact analysis work task. Adapted from Paper V. Cogwheels indicate an information seeking activity that could be supported by IR-based trace recovery.

ing based on textual similarity in the software engineering context [58]. Furthermore, work on trace link structures would enable us to explore the use of visualization techniques to support engineers' trace links seeking, as has previously been proposed by Cleland-Huang and Habrat [19].

We also suspect that other pieces of artifact meta-information could be useful in trace recovery. Web search engines consider hundreds of features to assess the relevance of web pages for ranking purposes [1]. Learning-to-rank methods are then used on training data to learn the optimal combination of feature weights, resulting in the best ranking of search results [67]. In the context of trace recovery, we envision that both nominal software artifact features (e.g., responsible team, subsystem), ordinal features (e.g., safety level, severity), and features measurable on a ratio scale (e.g, resolution time, link structure) can be used to improve ranking of candidate trace links, in particular when combined with information about the user of the tool. Engineers conducting trace recovery might not consider the relevance of candidate trace links to be binary, but rather of a multi-dimensional nature [61], i.e., dynamic and situational. For example, the relevance of a trace link might depend on the role of the tracing engineer (tester, developer, manager, etc.), the current phase of the development project (pre-study, implementation, verification, etc.), and which other trace links have already been identified (as there might be dependencies). Using meta-information and user information, IR-based trace recovery could assumably be advanced beyond what is possible using merely textual similarity analysis.

Figure 6: Linked structures of issue reports in the public Android issue tracking system.

We anticipate certain challenges as we continue our work. First, in many enterprises, information access is hindered by information being widely dispersed in information management systems with poor interoperability [69], resulting in what is referred to as *information silos*. It is uncertain which artifacts could be accessed without major engineering efforts and without breaking information access policies. Second, as identified by Klevin [64], the impact analysis reports in the targeted case, i.e., the answers to the template presented in Table 4, are stored in the issue tracking system as unstructured text. Clearly, this will complicate information extraction and data mining from the reports. Third, while the number of software artifacts in large projects can be challenging, it is several orders of magnitude smaller than the number of web pages indexed by modern web search engines. There is a risk that we will not be able to gather enough data for machine learning methods to do themselves justice.

9 Conclusion

The challenge of maintaining trace links in large-scale software engineering has been addressed by IR approaches in roughly a hundred previous publications. In an SLR, we identified 79 publications reporting empirical evaluations of IR-based trace recovery. We found that most often algebraic IR models have been applied

Future work	Description	Research method	Type of research
DT1	How can we further improve the trace link seeking model?	Design science	Modeling
FQ1	Is the trace link seeking model applicable in other development contexts with process requirements on traceability?	Multi-case study	Exploratory
DT2	How can textual similarity analysis be applied to support trace recovery in the impact analysis?	Design science	Tool development
FQ2	Can the accuracy of the tool output be improved by considering artifact meta-information?	Technology-oriented experiment	Evaluative
FQ3	Does the tool support the impact analysis work task?	Case study	Evaluative

Table 5: Future research questions and planned design science tasks.

(RQ1), and confirm the previous claim that no IR model regularly outperforms trace recovery based on VSM (RQ2).

A majority of previous evaluations of IR-based trace recovery have been technology-oriented, conducted in what Ingwersen and Järvelin refer to as "the cave of IR evaluation". Also, we show that evaluations of IR-based trace recovery primarily have been conducted using simplified datasets, both in relation to size (most often less than 500 artifacts) and origin (frequently former student projects, typically not validated for industrial representability). As such, the validity of concluding that IR-based trace recovery is feasible in an industrial setting, based on P-R curves "in the cave", can be questioned (RQ3).

On the other hand, a set of previous evaluations conducted with human subjects suggest that engineers would benefit from IR-based trace recovery tools when performing certain work tasks (RQ3). To further strengthen the evidence of IR-based trace recovery, more studies involving humans are needed, particularly industrial case studies (RQ5). Moreover, evaluative studies should be conducted on diverse datasets containing a higher number of artifacts (RQ4). Consequently, our findings intensify the call for additional empirical research by CoEST.

This thesis also includes two experiments on IR-based trace recovery. The technology-oriented experiment highlights the clear dependence between datasets and the accuracy of IR-based trace recovery, which was also confirmed by the SLR. Thus, to enable replications and secondary studies, we argue that datasets should be thoroughly characterized in future studies on trace recovery, especially when they cannot be disclosed (RQ4). The human-oriented experiment suggests that it

is worthwhile investigating further into the actual value of improved P-R curves. The pilot experiment showed that the effect size of using a slightly better tool is of practical significance regarding precision and F-measure. Finally, based on research on general IR evaluation, we propose a taxonomy of evaluation contexts tailored for IR-based trace recovery (RQ5).

As future work, we intend to target an industrial case of impact analysis in a safety-critical development context. In the case, engineers perform trace link seeking among textual artifacts, and explicitly specify the trace links according to a template. Consequently, the case appears to be suitable for evaluating IR-based trace recovery in an in-vivo setting.

BIBLIOGRAPHY

[1] E. Agichtein, E. Brill, and S. Dumais. Improving Web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, 2006.

[2] N. Ali, Y-G. Guéhéneuc, and G. Antoniol. Factors impacting the inputs of traceability recovery approaches. In J. Cleland-Huang, O. Gotel, and A. Zisman, editors, *Software and Systems Traceability*. Springer, 2012.

[3] M. Alvesson and K. Sköldberg. *Reflexive methodology: New vistas for qualitative research*. Sage Publications, 2000.

[4] G. Antoniol, G. Canfora, G. Casazza, and A. De Lucia. Information retrieval models for recovering traceability links between code and documentation. In *Conference on Software Maintenance*, pages 40–49, 2000.

[5] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering traceability links between code and documentation. In *Transactions on Software Engineering*, volume 28, pages 970–983, 2002.

[6] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval: The concepts and technology behind search*. Addison-Wesley, 2nd edition, 2011.

[7] L. Barker, E. Luman, M. McCauley, and S. Chu. Assessing equivalence: An alternative to the use of difference tests for measuring disparities in vaccination coverage. *American Journal of Epidemiology*, 156(11):1056–1061, 2002.

[8] E. Ben Charrada, D. Caspar, C. Jeanneret, and M. Glinz. Towards a benchmark for traceability. In *Proceedings of the 12th International Workshop on Principles on Software Evolution*, pages 21–30, 2011.

[9] J. Berlik, S. Dharmadhikari, M. Harding, and N. Singh. System and method for maintaining requirements traceability, United States Patent 8191044, 2012.

[10] D. Binkley and D. Lawrie. Information retrieval applications in software maintenance and evolution. In J. Marciniak, editor, *Encyclopedia of software engineering*. Taylor & Francis, 2nd edition, 2010.

[11] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3(4-5):993–1022, 2003.

[12] B. Boehm. Software engineering. 25(12):1226–1241, 1976.

[13] R. Burgin. The Monte Carlo method and the evaluation of retrieval system performance. *Journal of the American Society for Information Science*, 50(2):181–191, 1999.

[14] Carnegie Mellon Software Engineering Institute. CMMI for development, Version 1.3, 2010.

[15] D. Case. *Looking for information: A survey of research on information seeking, needs and behavior*. Academic Press, 2nd edition, 2007.

[16] J. Cleland-Huang, C. K Chang, and M. Christensen. Event-based traceability for managing evolutionary change. *Transactions on Software Engineering*, 29(9):796– 810, 2003.

[17] J. Cleland-Huang, A. Czauderna, A. Dekhtyar, O. Gotel, J. Huffman Hayes, E. Keenan, J. Maletic, D. Poshyvanyk, Y. Shin, A. Zisman, G. Antoniol, B. Berenbach, A. Egyed, and P. Mäder. Grand challenges, benchmarks, and TraceLab: Developing infrastructure for the software traceability research community. In *Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*, 2011.

[18] J. Cleland-Huang, O. Gotel, and A. Zisman, editors. *Software and systems traceability*. Springer, 2012.

[19] J. Cleland-Huang and R. Habrat. Visualization and analysis in automated trace retrieval. In *Proceedings of the 2nd International Workshop on Requirements Engineering Visualization*, 2007.

[20] C. Cleverdon. The significance of the Cranfield tests on index languages. In *Proceedings of the 14th Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1991.

[21] B. Dagenais, H. Ossher, R. Bellamy, M. Robillard, and J. de Vries. Moving into a new software project landscape. In *Proceedings of the 32nd International Conference on Software Engineering*, pages 275–284, 2010.

[22] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. ADAMS re-trace: A traceability recovery tool. In *Proceedings of the 9th European Conference on Software Maintenance and Reengineering*, pages 32–41, 2005.

[23] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Recovering traceability links in software artifact management systems using information retrieval methods. *Transactions on Software Engineering and Methodology*, 16(4), 2007.

[24] A. De Lucia, A. Marcus, R. Oliveto, and D. Poshyvanyk. Information retrieval methods for automated traceability recovery. In J. Cleland-Huang, O. Gotel, and A. Zisman, editors, *Software and Systems Traceability*. Springer, 2012.

[25] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[26] A. Dekhtyar and J. Huffman Hayes. Good benchmarks are hard to find: Toward the benchmark for information retrieval applications in software engineering. *Proceedings of the International Conference on Software Maintenance*, 2006.

[27] A. Dekhtyar, J. Huffman Hayes, and G. Antoniol. Benchmarks for traceability? In *Proceedings of the International Symposium on Grand Challenges in Traceability*, 2007.

[28] K. Desouza, Y. Awazu, and P. Baloh. Managing knowledge in global software development efforts: Issues and practices. *Software, IEEE*, 23(5):30–37, 2006.

[29] R. Dömges and K. Pohl. Adapting traceability environments to project-specific needs. *Communications of the ACM*, 41(12):54–62, 1998.

[30] M. Dorfman. *Standards, guidelines, and examples on system and software requirements engineering*. IEEE Computer Society Press, 1994.

[31] S. Easterbrook, J. Singer, M. Storey, and D. Damian. Selecting empirical methods for software engineering research. In F. Shull, J. Singer, and D. Sjöberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer, 2008.

[32] A. Edmunds and A. Morris. The problem of information overload in business organisations: A review of the literature. *International Journal of Information Management*, 20(1):17–28, 2000.

[33] M. Eppler and J. Mengis. The concept of information overload: A review of literature from organization science, accounting, marketing, MIS, and related disciplines. *The Information Society*, 20(5):325–344, 2004.

[34] D. Falessi, G. Cantone, and G. Canfora. A comprehensive characterization of NLP techniques for identifying equivalent requirements. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, 2010.

[35] D. Falessi, G. Cantone, and G. Canfora. Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. *Transactions on Software Engineering*, 2011.

[36] R. Fiutem and G. Antoniol. Identifying design-code inconsistencies in object-oriented software: A case study. In *Proceedings of the International Conference on Software Maintenance*, pages 94–102, 1998.

[37] L. Freund, E. Toms, and J. Waterhouse. Modeling the information behaviour of software engineers using a work - task framework. *Proceedings of the American Society for Information Science and Technology*, 42(1), 2005.

[38] L. Getoor and C. Diehl. Link mining: A survey. *SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.

[39] O. Gotel, J. Cleland-Huang, J. Huffman Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, J. Maletic, and P. Mäder. Traceability fundamentals. In J. Cleland-Huang, O. Gotel, and A. Zisman, editors, *Software and Systems Traceability*, pages 3–22. Springer, 2012.

[40] O. Gotel and C. Finkelstein. An analysis of the requirements traceability problem. In *Proceedings of the First International Conference on Requirements Engineering*, pages 94–101, 1994.

[41] S. Harter and C. Hert. Evaluation of information retrieval systems: Approaches, issues, and methods. *Annual Review of Information Science and Technology*, 32:3–94, 1997.

[42] D. Hawking. Challenges in enterprise search. In *Proceedings of the 15th Australasian database conference*, pages 15–24, 2004.

[43] A. Hevner, S. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004.

[44] Hewlett Packard Development Company. HP quality center software (formerly HP TestDirector for quality center software) Data sheet, 4AA0-9587ENW rev. 3, 2009.

[45] T. Hofman. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196, 2001.

[46] M Höst, B. Regnell, and C. Wohlin. Using students as subjects: A comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, 5(3):201–214, 2000.

[47] J. Huffman Hayes and A. Dekhtyar. A framework for comparing requirements tracing experiments. *Interational Journal of Software Engineering and Knowledge Engineering*, 15(5):751–781, 2005.

[48] J. Huffman Hayes, A. Dekhtyar, and S. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *Transactions on Software Engineering*, 32(1):4–19, 2006.

[49] J. Huffman Hayes, A. Dekhtyar, S. Sundaram, A. Holbrook, S. Vadlamudi, and A. April. REquirements TRacing on target (RETRO): improving software maintenance through traceability recovery. *Innovations in Systems and Software Engineering*, 3(3):193–202, 2007.

[50] IEEE Computer Society. 610.12-1990 IEEE Standard glossary of software engineering terminology. Technical report, 1990.

[51] P. Ingwersen and K. Järvelin. *The turn: Integration of information seeking and retrieval in context*. Springer, 2005.

[52] International Electrotechnical Commission. IEC 61511-1 ed 1.0, Safety instrumented systems for the process industry sector, 2003.

[53] International Electrotechnical Commission. IEC 61508 ed 2.0, Electrical/-Electronic/Programmable electronic safety-related systems, 2010.

[54] International Organization for Standardization. ISO 26262-1:2011 Road vehicles –Functional safety –, 2011.

[55] S. Isaac and W. Michael. *Handbook in research and evaluation: A collection of principles, methods, and strategies useful in the planning, design, and evaluation of studies in education and the behavioral sciences*. Edits Pub, 3rd edition, 1995.

[56] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48, 2000.

[57] A. Jedlitschka, M. Ciolkowski, and D. Pfahl. Reporting experiments in software engineering. In F. Shull, J. Singer, and D. Sjöberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 201–228. Springer, London, 2008.

[58] G. Karabatis, Z. Chen, V. Janeja, T. Lobo, M. Advani, M. Lindvall, and R. Feldmann. Using semantic networks and context in search for relevant software engineering artifacts. In S. Spaccapietra and L. Delcambre, editors, *Journal on Data Semantics XIV*, pages 74–104. Springer, Berlin, 2009.

[59] P. Karr-Wisniewski and Y. Lu. When more is too much: Operationalizing technology overload and exploring its impact on knowledge worker productivity. *Computers in Human Behavior*, 26(5):1061–1072, 2010.

[60] E. Keenan, A. Czauderna, G. Leach, J. Cleland-Huang, Y. Shin, E. Morlitz, M. Gethers, D. Poshyvanyk, J. Maletic, J. Huffman Hayes, A. Dekhtyar, A. Manukian, S. Hussein, and D. Hearn. TraceLab: an experimental workbench for equipping researchers to innovate, synthesize, and comparatively evaluate traceability solutions. In *Proceedings of the 34th International Conference on Software Engineering*, 2012.

[61] J. Kekäläinen and K. Järvelin. Evaluating information retrieval systems under the challenges of interaction and multidimensional dynamic relevance. *Proceedings of the COLIS 4 Conference*, pages 253—270, 2002.

[62] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. *EBSE Technical Report*, 2007.

[63] B. Kitchenham, T. Dybå, and M. Jörgensen. Evidence-based software engineering. In *Proceedings of the 26th International Conference on Software Engineering*, volume 2004, pages 273–281, 2004.

[64] A. Klevin. *People, process and tools: A study of impact analysis in a change process*. Master thesis, Lund University, ISSN 1650-2884, http://sam.cs.lth.se/ExjobGetFile?id=434, 2012.

[65] G. Leckie, K. Pettigrew, and C. Sylvain. Modeling the information seeking of professionals: A general model derived from research on engineers, health care professionals, and lawyers. *Library Quarterly*, 66(2):161–93, 1996.

[66] J. Lin, L. Chan, J. Cleland-Huang, R. Settimi, J. Amaya, G. Bedford, B. Berenbach, O. B Khadra, D. Chuan, and X. Zou. Poirot: A distributed tool supporting enterprise-wide automated traceability. In *Proceedings of the 14th International Conference on Requirements Engineering*, pages 363–364, 2006.

[67] T-Y Liu. *Learning to rank for information retrieval*. Springer, 2011.

[68] M. Lormans and A. van Deursen. Can LSI help reconstructing requirements traceability in design and test? In *Proceedings of the 10th European Conference on Software Maintenance and Reengineering*, pages 45–54, 2006.

[69] C. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[70] A. Marcus and J. Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. In *Proceedings of the International Conference on Software Engineering*, pages 125–135, 2003.

[71] P. Morville. *Ambient findability: What we find changes who we become*. O'Reilly Media, 2005.

[72] P. Naur and B. Randell, editors. *Software Engineering: Report of a conference sponsored by the NATO Science Committee*. 1969.

[73] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *International Conference on Program Comprehension*, pages 68–71, 2010.

[74] T. Olsson. *Software information management in requirements and test documentation*. Licentiate thesis, Lund University, 2002.

[75] R. Parasuraman, T. Sheridan, and C. Wickens. A model for types and levels of human interaction with automation. *Transactions on Systems, Man and Cybernetics*, 30(3):286–297, 2000.

[76] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, pages 71–80, 2008.

[77] A. Pinsonneault and K. Kraemer. Survey research methodology in management information systems: An assessment. *Journal of Management Information Systems*, 10(2):75–105, 1993.

[78] J. Ponte and B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, 1998.

[79] M. Poppendieck and T. Poppendieck. *Lean software development: An agile toolkit*. Addison-Wesley Professional, 2003.

[80] B. Ramesh. Process knowledge management with traceability. *IEEE Software*, 19(3):50–52, 2002.

[81] B. Randall. Towards a methodology of computing system design. In P. Naur and B. Randall, editors, *NATO Working Conference on Software Engineering 1968, Report on a Conference Sponsored by NATO Scientific Committee*, pages 204–208. 1969.

[82] S. E. Robertson and S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[83] B. Robson. *Real world research*. Blackwell, 2nd edition, 2002.

[84] P. Runeson, M. Höst, A. Rainer, and B. Regnell. *Case study research in software engineering: Guidelines and examples*. Wiley, 2012.

[85] G. Sabaliauskaite, A. Loconsole, E. Engström, M. Unterkalmsteiner, B. Regnell, P. Runeson, T. Gorschek, and R. Feldt. Challenges in aligning requirements engineering and verification in a large-scale industrial context. In *Requirements Engineering: Foundation for Software Quality*, pages 128–142, 2010.

[86] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Commununications of the ACM*, 18(11):613–620, 1975.

[87] W. Shaw Jr, R. Burgin, and P. Howell. Performance standards and evaluations in IR test collections: Vector-space and other retrieval models. *Information Processing & Management*, 33(1):15–36, 1997.

[88] F. Shull, J. Carver, S. Vegas, and N. Juristo. The role of replications in empirical software engineering. *Empirical Software Engineering*, 13(2):211–218, 2008.

[89] F. Shull, J. Singer, and D. Sjöberg. *Guide to advanced empirical software engineering*. Springer, 1st edition, 2010.

[90] K. Spärck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: Development and comparative experiments. *Information Processing and Management*, 36(6):779–808, 2000.

[91] R. Torkar, T. Gorschek, R. Feldt, M. Svahnberg, U. Raja, and K. Kamran. Requirements traceability: A systematic review and industry case study. *International Journal of Software Engineering and Knowledge Engineering*, 22(3):1–49, 2012.

[92] H. Turtle and B. Croft. Evaluation of an inference network-based retrieval model. *Transactions on Information Systems*, 9(3):187–222, 1991.

[93] J. Urbano. Information retrieval meta-evaluation: Challenges and opportunities in the music domain. In *International Society for Music Information Retrieval Conference*, pages 597–602, 2011.

[94] S. Wellek. *Testing statistical hypotheses of equivalence*. Chapman and Hall, 2003.

[95] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: A practical guide*. Springer, 2012.

[96] R. Yin. *Case study research: Design and methods*. Sage Publications, 3rd edition, 2003.

[97] H. Zantout and F. Marir. Document management systems from current capabilities towards intelligent information retrieval: An overview. *International Journal of Information Management*, 19(6):471–484, 1999.

PAPER I

RECOVERING FROM A DECADE: A SYSTEMATIC REVIEW OF INFORMATION RETRIEVAL APPROACHES TO SOFTWARE TRACEABILITY

Abstract

Context: Engineers in large-scale software development have to manage large amounts of information, spread across many artifacts. Several researchers have proposed expressing retrieval of trace links among artifacts, i.e., trace recovery, as an Information Retrieval (IR) problem. **Objective:** The objective of this study is to review work on IR-based trace recovery, with a particular focus on previous evaluations and strength of evidence. **Method:** We conducted a systematic literature review of IR-based trace recovery. **Results:** Of the 79 publications classified, a majority applied algebraic IR models. We found no IR model to regularly outperform the classic vector space model. While a set of studies on students indicate that IR-based trace recovery tools support certain work tasks, most previous studies do not go beyond reporting precision and recall of candidate trace links from evaluations using datasets containing less than 500 artifacts. **Conclusions:** We found a clear dependence between datasets and the accuracy of IR-based trace recovery. Also, our review identified a need of industrial case studies. Furthermore, we conclude that the overall quality of reporting should be improved regarding both context and tool details, measures reported, and use of IR terminology. Finally, based on our empirical findings, we present suggestions on how to advance research on IR-based trace recovery.

Markus Borg, Per Runeson, and Anders Ardö, *Submitted to a journal*, 2012

1 Introduction

The successful evolution of software systems involves concise and quick access to information. However, information overload plagues software engineers, as large amounts of formal and informal information is continuously produced and modified [36, 133]. Inevitably, especially in large-scale projects, this leads to a challenging information landscape, that includes, apart from the source code itself, requirements specifications of various abstraction levels, test case descriptions, defect reports, manuals, and the like. The state-of-practice approach to structure such information is to organize artifacts in databases, e.g., document management systems, requirements databases, and code repositories, and to manually maintain trace links [80, 88]. With access to trace information, engineers can more efficiently perform work tasks such as impact analysis, identification of reusable artifacts, and requirements validation [7, 169]. Furthermore, research has identified lack of traceability to be a major contributing factor in project overruns and failures [36, 69, 80]. Moreover, as traceability plays a role in software verification, safety standards such as ISO 26262 [94] for the automotive industry, and IEC 61511 [93] for the process industry, mandate maintenance of traceability information [99], as does the CMMI process improvement model [31]. However, manually maintaining trace links is an approach that does not scale [81]. In addition, the dynamics of software development makes it tedious and error-prone [69, 73, 88].

As a consequence, engineers would benefit from additional means of dealing with information seeking and retrieval, to navigate effectively the heterogeneous information landscape of software development projects. Several researchers have claimed it feasible to treat traceability as an information retrieval (IR) problem [7, 51, 88, 115, 120]. Also, other studies have reported that the use of semi-automated trace recovery reduces human effort when performing requirements tracing [49,54, 58, 88, 130]. The IR approach builds upon the assumption that if engineers refer to the same aspects of the system, similar language is used across different software artifacts. Thus, tools suggest trace links based on Natural Language (NL) content. During the first decade of the millennium, substantial research effort has been spent on tailoring, applying, and evaluating IR techniques to software engineering, but we found that a comprehensive overview of the field is missing. Such a secondary analysis would provide an evidence based foundation for future research, and advise industry practice [103]. As such, the gathered empirical evidence could be used to validate, and possibly intensify, the recent calls for future research by the traceability research community [79], organized by the Center of Excellence for Software Traceability (CoEST)[1]. Furthermore, it could assess the recent claims that applying more advanced IR models does not improve results [73, 132].

We have conducted a Systematic Mapping (SM) study [102, 136] that clusters publications on IR-based trace recovery. Then, as suggested by Kitchenham and Charters [103], we used the resulting map as a starting point for a Systematic Lit-

[1] www.coest.org

erature Review (SLR). SMs and SLRs are primarily distinguished by their driving Research Questions (RQ) [102], i.e., an SM identifies research gaps and clusters evidence to direct future research, while an SLR synthesizes empirical evidence on a specific RQ. The rigor of the methodologies is a key asset in ensuring a comprehensive collection of published evidence. We define our overall goals of this study in four RQs (RQ1 and RQ2 guided the SM, while RQ3 and RQ4 were subsequent starting points for the SLR):

RQ1 Which IR models and enhancement strategies have been most frequently applied to perform trace recovery among NL software artifacts?

RQ2 Which types of NL software artifacts have been most frequently linked in IR-based trace recovery studies?

RQ3 How strong is the evidence, wrt. degree of realism in the evaluations, of IR-based trace recovery?

RQ4 Which IR model has been the most effective, wrt. precision and recall, in recovering trace links?

This paper is organized as follows. Section 2 contains a thorough definition of the IR terminology we refer to throughout this paper, and a description of how IR tools can be used in a trace recovery process. Section 3 presents related work, i.e., the history of IR-based trace recovery, and related secondary and methodological studies. Section 4 describes how the SLR was conducted, including the collection of studies and the subsequent synthesis of empirical results. Section 4 shows the results from the study. Section 6 discusses our research questions based on the results. Finally, Section 7 presents a summary of our contributions and suggests directions for future research.

2 Background

This section presents fundamentals of IR, and how tools implementing IR models can be used in a trace recovery process.

2.1 IR background and terminology

As the study identified variations in use of terminology, this section defines the terminology used in this study (summarized in Table 1), which is aligned with recently redefined terms [39]. We use the following IR definition: "*information retrieval is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)*" [119]. If a retrieved document satisfies such a need, we consider it relevant. We solely consider text retrieval in the study, yet we follow

convention and refer to it as IR. In our interpretation, the starting point is that any approach that retrieves documents relevant to a query qualifies as IR. The terms Natural Language Processing (NLP) and Linguistic Engineering (LE) are used in a subset of the mapped publications of this study, even if they refer to the same IR techniques. We consider NLP and LE to be equivalent and borrow two definitions from Liddy [111]: "*NL text is text written in a language used by humans to communicate to one another*", and "*NLP is a range of computational techniques for analyzing and representing NL text*". As a result, IR (referring to a process solving a problem) and NLP (referring to a set of techniques) are overlapping. In contrast to the decision by Falessi *et al.* [73] to consistently apply the term NLP, we choose to use IR in this study, as we prefer to focus on the process rather than the techniques. While trace recovery truly deals with solutions targeting NL text, we prefer to primarily consider it as a problem of satisfying an information need.

Furthermore, a "*software artifact is any piece of information, a final or intermediate work product, which is produced and maintained during software development*" [106], e.g., requirements, design documents, source code, test specifications, manuals, and defect reports. To improve readability, we refer to such pieces of information only as 'artifacts'. Regarding traceability, we use two recent definitions: "*traceability is the potential for traces to be established and used*" and "*trace recovery is an approach to create trace links after the artifacts that they associate have been generated or manipulated*" [39]. In the literature, the trace recovery process is referred to in heterogeneous ways including traceability link recovery, inter-document correlation, document dependency/similarity detection, and document consolidation. We refer to all such approaches as *trace recovery*, and also use the term *links* without differentiating between dependencies, relations and similarities between artifacts.

In line with previous research, we use the term *dataset* to refer to the set of artifacts that is used as input in evaluations and *preprocessing* to refer to all processing of NL text before the IR models (discussed next) are applied [14], e.g., stop word removal, stemming and identifier (ID) splitting names expressed in CamelCase (i.e., identifiers named according to the coding convention to capitalize the first character in every word) or identifiers named according to the under_score convention. *Feature selection* is the process of selecting a subset of terms to represent a document, in an attempt to decrease the size of the effective vocabulary and to remove noise [119].

To support trace recovery, several IR models have been applied. Since we identified contradicting interpretations of what is considered a model, weighting scheme, and similarity measure, we briefly present our understanding of the IR field. IR models often apply the *bag-of-words model*, a simplifying assumption that represents a document as an unordered collection of words, disregarding word order [119]. Most existing IR models can be classified as either algebraic or probabilistic, depending on how relevance between queries and documents is measured. In algebraic IR models, relevance is assumed to be correlated with

similarity [173]. The most well-known algebraic model is the commonly applied *Vector Space Model (VSM)* [150], which due to its many variation points acts as a framework for retrieval. Common to all variations of VSM is that both documents and queries are represented as vectors in a high-dimensional space (every term, after preprocessing, in the document collection constitutes a dimension) and that similarities are calculated between vectors using some distance function. Individual terms are not equally meaningful in characterizing documents, thus they are weighted accordingly. Term weights can be both binary (i.e., existing or non-existing) and raw (i.e., based on term frequency) but usually some variant of *Term Frequency-Inverse Document Frequency* (TF-IDF) weighting is applied. TF-IDF is used to weight a term based on the length of the document and the frequency of the term, both in the document and in the entire document collection [154]. Regarding *similarity measures*, the cosine similarity (calculated as the cosine of the angle between vectors) is dominating in IR-based trace recovery using algebraic models, but also Dice's coefficient and the Jaccard index [119] have been applied. In an attempt to reduce the noise of NL (such as synonymy and polysemy), *Latent Semantic Indexing* (LSI) was introduced [60]. LSI reduces the dimensions of the vector space, finding semi-dimensions using singular value decomposition. The new dimensions are no longer individual terms, but concepts represented as combinations of terms. In the VSM, *relevance feedback* (i.e., improving the query based on human judgement of partial search results, followed by re-executing an improved search query) is typically achieved by updating the query vector [173]. In IR-based trace recovery, this is commonly implemented using the Standard Rocchio method [145]. The method adjusts the query vector toward the centroid vector of the relevant documents, and away from the centroid vector of the non-relevant documents.

In probabilistic retrieval, relevance between a query and a document is estimated by probabilistic models. The IR is expressed as a classification problem, documents being either relevant or non-relevant [154]. Documents are then ranked according to their probability of being relevant [124], referred to as the probabilistic ranking principle [141]. In trace recovery, the *Binary Independence Retrieval* model (BIM) [144] was first applied to establish links. BIM naïvely assumes that terms are independently distributed, and essentially applies the Naïve Bayes classifier for document ranking [109]. Different weighting schemes have been explored to improve results, and currently the *BM25* weighting used in the non-binary Okapi system [143] constitutes state-of-the-art.

Another category of probabilistic retrieval is based on the model of an inference process in a *Probabilistic Inference Network* (PIN) [164]. In an inference network, relevance is modeled by the uncertainty associated with inferring the query from the document [173]. Inference networks can embed most other IR models, which simplifies the combining of approaches. In its simplest implementation, a document instantiates every term with a certain strength and multiple terms accumulate to a numerical score for a document given each specific query. Relevance

feedback is possible also for BIM and PIN retrieval [173], but we have not identified any such attempts within the trace recovery research.

In the last years, another subset of probabilistic IR models has been applied to trace recovery. *Statistical Language Models* (LM) estimate an LM for each document, then documents are ranked based on the probability that the LM of a document would generate the terms of the query [138]. A refinement of simple LMs, topic models, describes documents as a mixture over topics. Each individual topic is then characterized by an LM [174]. In trace recovery research, studies applying the four topic models *Probabilistic Latent Semantic Indexing* (PLSI) [82], *Latent Dirichlet Allocation* (LDA) [20], *Correlated Topic Model* (CTM) [19] and *Relational Topic Model* (RTM) [33] have been conducted. To measure the distance between LMs, where documents and queries are represented as stochastic variables, several different measures of distributional similarity exist, such as the *Jensen-Shannon divergence* (JS). To the best of our knowledge, the only implementation of relevance feedback in LM-based trace recovery was based on the *Mixture Model method* [175].

Finally, a number of measures used to evaluate IR tools need to be defined. Accuracy of a set of search results is primarily measured by the standard IR-measures *precision* (the fraction of retrieved instances that are relevant), *recall* (the fraction of relevant instances that are retrieved) and *F-measure* (harmonic mean of precision and recall, possibly weighted to favour one over another) [14]. Precision and recall values (P-R values) are typically reported pairwise or as precision and recall curves (P-R curves). Two other set-based measures, originating from the traceability community, are *Recovery Effort Index* (REI) [7] and *Selectivity* [162]. *Secondary measures* aim to go further than comparing sets of search results, and also consider their internal ranking. Two standard IR measures are *Mean Average Precision* (MAP) of precision scores for a query [119], and *Discounted Cumulative Gain* (DCG) [95] (a graded relevance scale based on the position of a document among search results). To address this matter in the specific application of trace recovery, Sundaram *et al.* [162] proposed *DiffAR*, *DiffMR*, and *Lag* to assess the quality of retrieved candidate links.

2.2 IR-based support in a trace recovery process

As the candidate trace links generated by state-of-the-art IR-based trace recovery typically are too inaccurate, the current tools are proposed to be used in a semi-automatic process. De Lucia *et al.* describe this process as a sequence of four key steps, where the fourth step requires human judgement [55]. Although steps 2 and 3 mainly apply to algebraic IR models, also other IR models can be described by a similar sequential process flow. The four steps are:

1. document parsing, extraction, and pre-processing
2. corpus indexing with an IR method

Retrieval Models			Misc.		
Algebraic models	**Probabilistic models**	**Statistical language models**	**Weighting schemes**	**Similarity measures / distance functions**	**Relevance feedback models**
Vector Space Model (VSM)	Binary Independence Model (BIM)	Language Model (LM)	Binary	Cosine similarity	Standard Rochio
Latent Semantic Indexing (LSI)	Probabilistic Inference Network (PIN)	Probabilistic Latent Semantic Indexing (PLSI)	Raw	Dice's coefficient	Mixture Model
	Best Match 25 (BM25)[a]	Latent Dirichlet Allocation (LDA)	Term Frequency Inverse Document Frequency (TFIDF)	Jaccard index	
		Correlated Topics Model (CTM)	Best Match 25 (BM25)[a]	Jensen-Shannon divergence (JS)	
		Relational Topics Model (RTM)			

[a] Okapi BM25 is used to refer both to a non-binary probabilistic model, and its weighting scheme.

Table 1: A summary of fundamental IR terms applied in trace recovery. Note that only the vertical organization carries a meaning.

3. ranked list generation
4. analysis of candidate links

In the first step, the artifacts in the targeted information space are processed and represented as a set of documents at a given granularity level, e.g., sections, class files or individual requirements. In the second step, for algebraic IR models, features from the set of documents are extracted and weighted to create an index. When also the query has been indexed in the same way, the output from step 2 is used to calculate similarities between artifacts to rank candidate trace links accordingly. In the final step, these candidate trace links are provided to an engineer for examination. Typically, the engineer then reviews the candidate source and target artifacts of every candidate trace link, and determines whether the link should be confirmed or not. Consequently, the final outcome of the process of IR-based trace recovery is based on human judgment.

A number of publications propose advice for engineers working with candidate trace links. De Lucia *et al.* have suggested that an engineer should iteratively decrease the similarity threshold, and stop considering candidate trace links when the fraction of incorrect links get too high [56, 57]. Based on an experiment with student subjects, they concluded that an incremental approach in general both improves the accuracy and reduces the effort involved in a tracing task supported by IR-based trace recovery. Furthermore, they report that the subjects preferred working in an incremental manner. Working incrementally with candidate trace links can to some subjects also be an intuitive approach. In a previous experiment by Borg and Pfahl, several subjects described such an approach to deal with tool output, even without explicit instructions [22]. Coverage analysis is another strategy proposed by De Lucia *et al.*, intended to follow up on the step of iteratively decreasing the similarity threshold [59]. By analyzing the confirmed candidate trace links, i.e., conducting a coverage analysis, De Lucia *et al.* suggest that engineers should focus on tracing artifacts that have few trace links. Also, in an experiment with students, they demonstrated that an engineer working according to this strategy recovers more correct trace links.

3 Related work

This section presents a chronological overview of IR-based trace recovery, previous overviews of the field, and related work on advancing empirical evaluations of IR-based trace recovery.

3.1 A brief history of IR-based trace recovery

Tool support for the linking process of NL artifacts has been explored by researchers since at least the early 1990s. Pioneering work was done in the LESD

project (Linguistic Engineering for Software Design) by Borillo *et al.*, in which a tool suite analyzing NL requirements was developed [25]. The tool suite parsed NL requirements to build semantic representations, and used artificial intelligence approaches to help engineers establish trace links between individual requirements [26]. Apart from analyzing relations between artifacts, the tools evaluated consistency, completeness, verifiability and modifiability [32]. In 1998, a study by Fiutem and Antoniol presented a recovery process to bridge the gap between design and code, based on edit distances between artifacts [75]. They coined the term "traceability recovery", and Antoniol *et al.* published several papers on the topic. Also, they were the first to clearly express identification of trace links as an IR problem [5]. Their milestone work from 2002 compared two standard IR models, probabilistic retrieval using the BIM and the VSM [7]. Simultaneously, in the late 1990s, Park *et al.* worked on tracing dependencies between artifacts using a sliding window combined with syntactic parsing [134]. Similarities between sentences were calculated using cosine similarities.

During the first decade of the new millennium, several research groups advanced IR-based trace recovery. Natt och Dag *et al.* did research on requirement dependencies in the dynamic environment of market-driven requirements engineering [129]. They developed the tool ReqSimile, implementing trace recovery based on the VSM, and later evaluated it in a controlled experiment [130]. A publication by Marcus and Maletic, the second most cited article in the field, constitutes a technical milestone in IR-based trace recovery [120]. They introduced Latent Semantic Indexing (LSI) to recover trace links between source code and NL documentation, a technique that has been used by multiple researchers since. Huffman Hayes and Dekhtyar enhanced VSM retrieval with relevance feedback and introduced secondary performance metrics [90]. From early on, their research had a human-oriented perspective, aimed at supporting V&V activities at NASA using their tool RETRO [89].

De Lucia *et al.* have conducted work focused on empirically evaluating LSI-based trace recovery in their document management system ADAMS [52]. They have advanced the empirical foundation by conducting a series of controlled experiments and case studies with student subjects [53, 54, 58]. Cleland-Huang and colleagues have published several studies on IR-based trace recovery. They introduced probabilistic trace recovery using a PIN-based retrieval model, implemented in their tool Poirot [112]. Much of their work has focused on improving the accuracy of their tool by enhancements such as: applying a thesaurus to deal with synonymy [152], extraction of key phrases [180], and using a project glossary to weight the most important terms higher [180].

Recent work on IR-based trace recovery has, with various results, gone beyond the traditional models for information retrieval. In particular, trace recovery supported by probabilistic topic models has been explored by several researchers. Dekhtyar *et al.* combined several IR models using a voting scheme, including the probabilistic topic model Latent Dirachlet Allocation (LDA) [65]. Parvathy *et al.*

proposed using the Correlated Topic Model (CTM) [135], and Gethers *et al.* suggested using Relational Topic Model (RTM) [77]. Abadi *et al.* proposed using Probabilistic Latent Semantic Indexing (PLSI) and utilizing two concepts based on information theory, Sufficient Dimensionality Reduction (SDR) and Jensen-Shannon Divergence (JS) [1]. Capobianco *et al.* proposed representing NL artifacts as B-splines and calculating similarities as distances between them on the Cartesian plane [30]. Sultanov and Huffman Hayes implemented trace recovery using a swarm technique [160], an approach in which a non-centralized group of non-intelligent self-organized agents perform work that, when combined, enables conclusions to be drawn.

3.2 Previous overviews on IR-based trace recovery

Basically every publication on IR-based trace recovery contains some information on previous research in the field. In our opinion, the previously most comprehensive summary of the field was provided by De Lucia *et al.* [58]. Even though the summary was not the primary contribution of the publication, they chronologically described the development, presented 15 trace recovery methods and 5 tool implementations. They compared underlying IR models, enhancing strategies, evaluation methodologies and types of recovered links. However, regarding both methodological rigor and depth of the analysis, it is not a complete SLR. De Lucia *et al.* have also surveyed proposed approaches to traceability management for impact analysis [50]. They discussed previous work based on a conceptual framework by Bianchi *et al.* [17], consisting of the three traceability dimensions: type of links, source of information to derive links, and their internal representation. Apart from IR-based methods, the survey by De Lucia *et al.* contains both rule-based and data mining-based trace recovery. Also Binkley and Lawrie have presented a survey of IR-based trace recovery as part of an overview of applications of IR in software engineering [18]. They concluded that the main focus of the research has been to improve the accuracy of candidate links wrt. P-R values, and that LSI has been the most popular IR model. However, they also report that no IR model has been reported as superior for trace recovery. While our work is similar to previous work, our review is more structured and goes deeper with a more narrow scope.

Another set of publications has presented taxonomies on IR techniques in software engineering. In an attempt to harmonize the terminology of the IR applications, Canfora and Cerulo presented a taxonomy of IR models [28]. However, their surveyed IR applications are not explicitly focusing on software engineering. Furthermore, their proposed taxonomy does not cover recent IR models identified in our study, and the subdivision into 'representation' and 'reasoning' poorly serves our intentions. Falessi *et al.* recently published a comprehensive taxonomy of IR techniques available to identify equivalent requirements [73]. They adopted the term variation point from Software Product Line Engineering [137], to stress

the fact that an IR solution is a combination of different, often orthogonal, design choices. They consider an IR solution to consist of a combination of algebraic model, term extraction, weighting scheme and similarity metric. Finally, they conducted an empirical study of various combinations and concluded that simple approaches yielded the most accurate results on their dataset. We share their view on variation points, but fail to apply it since our literature review is limited by what previous publications report on IR-based trace recovery. Also, their proposed taxonomy only covers algebraic IR models, excluding other models (most importantly, the entire family of probabilistic retrieval). The main difference between our study and the work by Falessi *et al.* is that they identified variation points of IR approaches and found the best combination by experimentation. Our study on the other hand systematically extracts empirical evidence from previous research, using the systematic literature approach, and aggregates results in a secondary analysis. However, since our IR taxonomies partly overlap, empirical results from the two studies can be compared.

Concept location (a.k.a. feature location) is a research topic that overlaps trace recovery. It can be seen as the first step of a change impact analysis process [122]. Given a concept (or feature) that is to be modified, the initial information need of a developer is to locate the part of the source code where it is embedded. Clearly, this information need could be fulfilled by utilizing IR. However, we distinguish the topics by considering concept location to be more query-oriented [76]. Furthermore, whereas trace recovery typically is evaluated by linking n artifacts to m other artifacts, evaluations of concept location tend to focus on n queries targeting a document set of m source code artifacts (where $n << m$), as for example in the study by Torchiano and Ricca [163]. Also, while it is often argued that trace recovery should retrieve trace links with a high recall, the goal of concept location is mainly to retrieve one single location in the code with high precision. Dit *et al.* recently published a literature review on feature location [68].

3.3 Related contributions to the empirical study of IR-based trace recovery

A number of previous publications have aimed at structuring or advancing the research on IR-based trace recovery, and are thus closely related to our study. An early attempt to advance reporting and conducting of empirical experiments was published by Huffman Hayes and Dekhtyar [84]. Their experimental framework describes the four phases: *definition*, *planning*, *realization* and *interpretation*. In addition, they used their framework to characterize previous publications. Unfortunately, the framework has not been applied frequently and the quality of the reporting of empirical evaluations varies greatly [24]. Huffman Hayes *et al.* also presented the distinction between studies of methods (are the tools capable of providing accurate results fast?) and studies of human analysts (how do humans use the tool output?) [88]. Furthermore, they proposed assessing the accuracy of tool

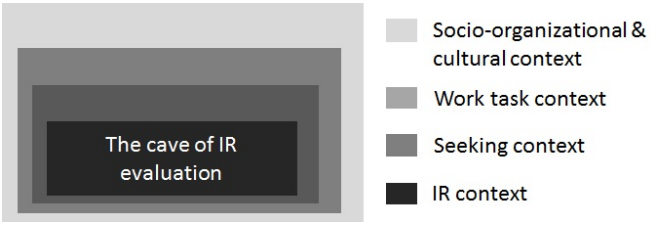


Figure 1: The Integrated Cognitive Research Framework by Ingwersen and Järvelin [92], a framework for IR evaluations in context.

output according to quality intervals named 'acceptable', 'good', and 'excellent', based on Huffman Hayes' industrial experience of working with traceability matrices of various qualities. Huffman Hayes et al.'s quality levels were defined to represent the effort that would be required by an engineer to vet an entire candidate traceability matrix. In our study, we use both classification schemes proposed by Huffman Hayes *et al.* to map the identified primary publications.

Considering empirical evaluations, we extend the classifications proposed by Huffman Hayes *et al.* [88] by an adapted version of the *Integrated Cognitive Research Framework* by Ingwersen and Järvelin [92]. Their work aimed at extending the de-facto standard of IR evaluation, the *Laboratory Model of IR Evaluation*, developed in the Cranfield tests in the 60s [44], challenged for its unrealistic lack of user involvement [101]. Ingwersen and Järvelin argued that IR is always evaluated in a context, referred to the innermost context as "the cave of IR evaluation", and proposed a framework consisting of four integrated contexts (see Figure 1). We have adapted their framework to a four-level context taxonomy, tailored for IR-based trace recovery, to classify in which contexts previous evaluations have been conducted, see Table 2. Also, we add a dimension of study environments (university, proprietary, and open source environment), as presented in Figure 12 in Section 4. For more information on the context taxonomy, we refer to our original publication [23].

In the field of IR-based trace recovery, the empirical evaluations are termed very differently by different authors. Some call them 'experiments', others 'case studies', and yet others only 'studies'. We use the following definitions, which are established in the field of software engineering.

Case study in software engineering is an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified. [147]

Experiment (or controlled experiment) in software engineering is an empirical enquiry that manipulates one factor or variable of the studied setting. Based

> in randomization, different treatments are applied to or by different subjects, while keeping other variables constant, and measuring the effects on outcome variables. In human-oriented experiments, humans apply different treatments to objects, while in technology-oriented experiments, different technical treatments are applied to different objects. [170]

Empirical evaluations of IR-based trace recovery may be classified as case studies, if they evaluate the use of, e.g., IR-based trace recovery tools in a complex software engineering environment, where it is not clear whether the tool is the main factor or other factors are at play. These are typically level 4 studies in our taxonomy, see Table 2. Human-oriented controlled experiments may evaluate human performance when using two different IR-tools in an artificial (in vitro) or well-controlled real (in vivo) environment, typically at level 3 of the taxonomy. The stochastical variation is here primarily assumed to be in the human behavior, although there of course are interactions between the human behavior, the artifacts and the tools. Technology-oriented controlled experiments evaluate tool performance on different artifacts, without human intervention, corresponding to levels 1 and 2 in our taxonomy. The variation factor is here the artifacts, and hence the technology-oriented experiment may be seen as benchmarking studies, where one technique is compared to another technique, using the same artifacts, or the performance of one technique is compared for multiple different artifacts.

The validity of the datasets used as input in evaluations in IR-based trace recovery is frequently discussed in the literature. Also, two recent publications primarily address this issue. Ali *et al.* present a literature review on characteristics of artifacts reported to impact trace recovery evaluations [4], e.g., ambiguous and vague requirements, and the quality of source code identifiers. Ali *et al.* extracted P-R values from eight previous trace recovery evaluations, not limited to IR-based trace recovery, and show that the same techniques generate candidate trace links of very different accuracy across datasets. They conclude that research targeting only recovery methods in isolation is not expected to lead to any major breakthroughs, instead they suggest that factors impacting the input artifacts should be better controlled. Borg *et al.* recently highlighted that a majority of previous evaluations of IR-based trace recovery have been conducted using artifacts developed by students [24]. The authors explored this potential validity threat in a survey of the traceability community. Their results indicate that while most authors consider artifacts originating from student projects to be only partly representative to industrial artifacts, few respondents explicitly validated them before using them as experimental input.

3.4 Precision and recall evaluation styles for technology-oriented trace recovery

In the primary publications, two principally different styles to report output from technology-oriented experiments have been used, i.e., presentation of P-R val-

Level 1: Retrieval context	The most simplified context, referred to as "the cave of IR evaluation". A strict retrieval context, performance is evaluated wrt. the accuracy of a set of search results. Quantitative studies dominate.	Precision, recall, F-measure	Experiments on benchmarks, possibly with simulated feedback
Level 2: Seeking context	A first step towards realistic applications of the tool, "drifting outside the cave'. A seeking context with a focus on how the human finds relevant information in what was retrieved by the system. Quantitative studies dominate.	Secondary measures. General IR: MAP, DCG. Traceability specific: Lag, DiffAR, DiffMR.	Experiments on benchmarks, possibly with simulated feedback
Level 3: Work task context	Humans complete real tasks, but in an in-vitro setting. Goal of evaluation is to assess the casual effect of an IR tool when completing a task. A mix of quantitative and qualitative studies.	Time spent on task and quality of work.	Controlled experiments with human subjects.
Level 4: Project context	Evaluations in a social-organizational context. The IR tool is studied when used by engineers within the full complexity of an in-vivo setting. Qualitative studies dominate.	User satisfaction, tool usage	Case studies

Table 2: A context taxonomy of IR-based trace recovery evaluations. Level 1 is technology-oriented, and level 3 and 4 are human-oriented. Level 2 typically has a mixed focus.

ues from evaluations in the retrieval and seeking contexts. A number of publications, including the pioneering work by Antoniol *et al.* [7], used the traditional style from the ad hoc retrieval task organized by the Text REtrieval Conference (TREC) [166], driving large-scale evaluations of IR. In this style, a number of queries are executed on a document set, and each query results in a ranked list of search results (cf. (a) in Figure 2). The accuracy of the IR system is then calculated as an average of the precision and recall over the queries. For example, in Antoniol *et al.*'s evaluation, source code files were used as queries and the document set consisted of individual manual pages. We refer to this reporting style as *query-based* evaluation. This setup evaluates the IR problem: "given this trace artifact, to which other trace artifacts should trace links be established?" The IR problem is reformulated for each trace artifact used as a query, and the results can be presented as a P-R curve displaying the average accuracy of candidate trace links over n queries. This reporting style shows how accurately an IR-based trace recovery tool supports a work task that requires single on-demand tracing efforts (a.k.a. reactive tracing or just-in-time tracing), e.g., establishing traces as part of an impact analysis work task [7, 22, 110].

In the other type of reporting style used in the primary publications, documents of different types are compared to each other, and the result from the similarity- or probability-based retrieval is reported as one single ranked list of candidate

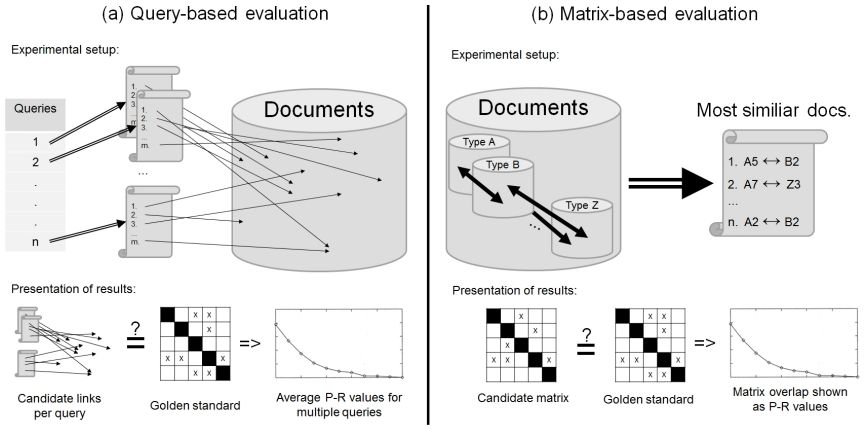


Figure 2: Query-based evaluation vs. matrix-based evaluation of IR-based trace recovery.

trace links. This can be interpreted as the IR problem: "among all these possible trace links, which trace links should be established?" Thus, the outcome is an entire candidate traceability matrix. We refer to this reporting style as *matrix-based* evaluation. The candidate traceability matrix can be compared to a golden standard, and the accuracy (i.e., overlap between the matrices) can be presented as a P-R curve, as shown in b) in Figure 2. This evaluation setup has been used in several primary publications to assess the accuracy of candidate traceability matrices generated by IR-based trace recovery tools. Also, Huffman Hayes *et al.* defined the quality intervals described in Section 3.3 to support this evaluation style [88].

Consequently, since the P-R values reported from query-based evaluations and matrix-based evaluations carry different meanings, the differences in reporting styles have to be considered when synthesizing results. Unfortunately, the primary publications do not always clearly report which evaluation style that has been used.

Apart from the principally different meaning of reported P-R values, the primary publications also differ by which sets of P-R values are reported. Precision and recall are set-based measures, and the accuracy of a set of candidate trace links (or candidate trace matrix) depends on which links are considered the tool output. Apart from the traditional way of reporting precision at fixed levels of recall, further described in Section 4.3, different strategies for selecting subsets of candidate trace links have been proposed. Such heuristics can be used by engineers working with IR-based trace recovery tools, and several primary publications report corresponding P-R values. We refer to these different approaches to consider subsets of ranked candidate trace links as *cut-off strategies*. Example cut-off strategies include: Constant cut point, a fixed number of the top-ranked trace links are selected,

e.g. 5, 10, or 50. Variable cut point, a fixed percentage of the total number of candidate trace links is selected, e.g. 5% or 10%. Constant threshold, all candidate trace links representing similarities (or probabilities) above a specified threshold is selected, e.g. above a cosine similarity of 0.7.

The choice of what subset of candidate trace links to represent by P-R values reflects the cut-off strategy an imagined engineer could use when working with the tool output. However, which strategy results in the most accurate subset of trace links depends on the specific case evaluated. Moreover, in reality it is possible that engineers would not be consistent in how they work with candidate trace links. As a consequence of the many possibly ways to report P-R values, the primary publications view output from IR-based trace recovery tools from rather different perspectives. For work tasks supported by a separate list of candidate trace links per source artifact, there are indications that human subjects seldom consider more than 10 candidate trace links [22], in line with what is commonplace to present as a 'pages-worth' output of major search engines such as Google, Bing and Yahoo. On the other hand, when an IR-based trace recovery tool is used to generate a candidate traceability matrix over an entire information space, considering only the first 10 candidate links would obviously be insufficient, as there would likely be thousands of correct trace links to recover. However, regardless of reporting style, the number of candidate trace links a P-R value represents is important in any evaluation of IR-based trace recovery tools, since a human is intended to vet the output.

4 Method

The overall goal of this study was to form a comprehensive overview of the existing research on IR-based trace recovery. To achieve this objective, we systematically collected empirical evidence to answer research questions characteristic both for an SM and an SLR [103, 136]. The study was conducted in the following distinct steps, (i) development of the review protocol, (ii) selection of publications, (iii) data extraction and mapping of publications, which were partly iterated and each of them was validated.

4.1 Protocol development

Following the established review guidelines for SLRs in software engineering [103], we iteratively developed a review protocol in consensus meetings between the authors. The protocol defined the research questions (stated in Section 1), the search strategy (described in Section 4.2), the inclusion/exclusion criteria (presented in Table 3), and the classification scheme used for the data extraction (described in Section 4.3). Also, the protocol specified use of qualitative cross-case analysis to synthesize data extracted from the primary studies [126], sometimes however extended by pieces of quantitative information. The extracted data were organized

in a tabular format to support comparison across studies. Evidence was summarized per category, and commonalities and differences between studies were investigated. Also, the review protocol specified the use of Zotero[2] as the reference management system, to simplify general tasks such as sorting, searching and removal of duplicates. An important deviation from the terminology used in the SLR guidelines is that we distinguish between *primary publications* (i.e., included units of publication) and *primary studies* (i.e., included pieces of empirical evidence), since a number of publications report multiple studies.

Table 3 states our inclusion/exclusion criteria, along with rationales and examples. A number of general decisions accompanied the criteria:

- Empirical results presented in several articles, we only included from the most extensive publication. Examples of excluded publications include pioneering work later extended to journal publications, the most notable being work by Antoniol *et al.* [5] and Marcus and Maletic [120]. However, we included publications describing all *independent replications* (deliberate variations of one or more major aspects), and *dependant replications* (same or very similar experimental setups) by other researchers [153].

- Our study included publications that apply techniques in E2a-d in Table 3, but use an IR model as benchmark. In such cases, we included the IR benchmark, and noted possible complementary approaches as enhancements. An example is work using probabilistic retrieval enhanced by machine learning from existing trace links [66].

- We included approaches that use structured NL as input, i.e., source code or tabular data, but treat the information as unstructured. Instead, we considered any attempts to utilize document structure as enhancements.

- Our study only included linking between software artifacts, i.e., artifacts that are produced and maintained during development [106]. Thus, we excluded linking approaches to entities such as e-mails [13] and tacit knowledge [83, 159].

- We excluded studies evaluating trace recovery in which neither the source nor the target artifacts dominantly represent information as NL text. Excluded publications comprise linking source code to test code [165], and work linking source code to text expressed in specific modelling notation [9, 41].

4.2 Selection of publications

The systematic identification of publications consisted of two main phases: (i) development of a golden standard of primary publications, and (ii) a search string that

[2] www.zotero.org

	Inclusion criteria	**Rationale/comments**
I1	Publication available in English in full text	We assumed that all relevant publications would be available in English.
I2	Publication is a peer-reviewed piece of software engineering work	As a quality assurance, we did not include technical reports, master theses etc.
I3	Publication contains empirical results (case study, experiment, survey etc.) of IR-based trace recovery where natural language artifacts are either source or target	Defined our main scope based on our RQs. Publication should clearly link artifacts, thus we excluded tools supporting a broader sense of program understanding such as COCONUT [48]. Also, the approach should treat the linking as an IR problem. However, we excluded solutions exclusively extracting specific character sequences in NL text, such as work on Mozilla defect reports [12].
	Exclusion criteria	**Rationale/comments**
E1	Answer is no to I1, I2 or I3	
E2	Publication proposes one of the following approaches to recover trace links, rather than IR: a) rule-based extraction b) ontology-based extraction c) machine learning approaches that require supervised learning d) dynamic/execution analysis	We included only publications that are deployable in an industrial setting with limited effort. Thus, we limited our study to techniques that require nothing but unstructured NL text as input. Other approaches could arguably be applied to perform IR, but are too different to fit our scope. Excluded approaches include: rules [71,157], ontologies [10], supervised machine learning [156], semantic networks [113], and dynamic analysis [72].
E3	Article explicitly targets one of the following topics, instead of trace recovery: a) concept/feature location b) duplicate/clone detection c) code clustering d) class cohesion e) cross cutting concerns/aspect mining	We excluded both concept location and duplicate detection since it deals with different problems, even if some studies apply IR models. Excluded publications include: duplicate detection of defects [146], detection of equivalent requirements [73], and concept location [122]. We explicitly added the topics code clustering, class cohesion, and cross cutting concerns to clarify our scope.

Table 3: Inclusion/exclusion criteria applied in our study. The rightmost column motivates our decisions.

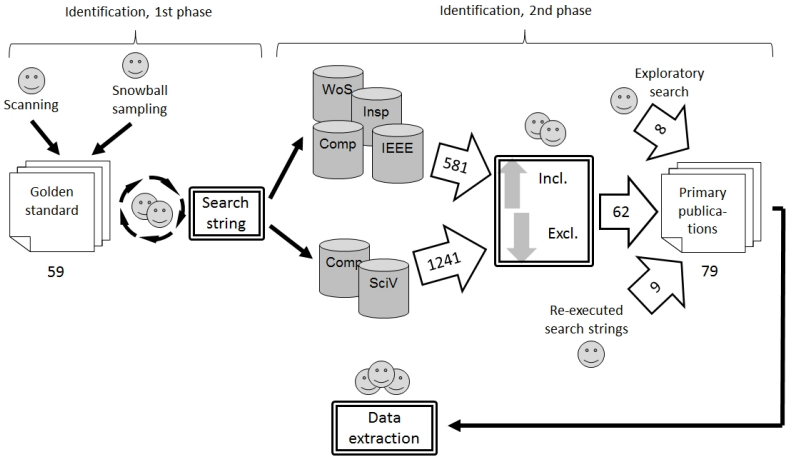


Figure 3: Overview of the publication selection phase. Smileys show the number of people involved in a step, while double frames represent a validation. Numbers refer to number of publications.

retrieves them, and a systematic search for publications, as shown in Figure 1. In the first phase, a set of publications was identified through exploratory searching, mainly by snowball sampling from a subset of an informal literature review. The most frequently recurring publication fora were then scanned for additional publications. This activity resulted in 59 publications, which was deemed our golden standard[3]. The first phase led to an understanding of the terminology used in the field, and made it possible to develop valid search terms.

The second step of the first phase consisted of iterative development of the search string. Together with a librarian at the department, we repeatedly evaluated our search string using combined searches in the Inspec/Compendex databases. Fifty-five papers in the golden standard were available in those databases. We considered the search string good enough when it resulted in 224 unique hits with 80% recall and 20% precision when searching for the golden standard, i.e., 44 of the 55 primary publications plus 176 additional publications were retrieved.

The final search string was composed of four parts connected with ANDs, specifying the *activity*, *objects*, *domain*, and *approach* respectively.

[3]The golden standard was not considered the end goal of our study, but was the target during the iterative development of the search string described next.

Primary Databases	**Search options**	**#Search results**
Inspec	Title+abstract, no auto-stem	194
Compendex	Title+abstract, no auto-stem	143
IEEE Explore	All fields	136
Web of Science	Title+abstract+keywords	108
Secondary Databases	**Search options**	**#Search results**
ACM Digital Library	All fields, auto-stem	1038
SciVerse Hub Beta	Science Direct+SCOPUS	203

Table 4: Search options used in databases, and the number of search results.

```
(traceability OR "requirements tracing" OR "requirements trace" OR
 "trace retrieval")
AND
(requirement* OR specification* OR document OR documents OR
 design OR code OR test OR tests OR defect* OR artefact* OR
 artifact* OR link OR links)
AND
(software OR program OR source OR analyst)
AND
("information retrieval" OR IR OR linguistic OR lexical OR
 semantic OR NLP OR recovery OR retrieval)
```

The search string was first applied to the four databases supporting export of search results to BibTeX format, as presented in Table 4. The resulting 581 papers were merged in Zotero. After manual removal of duplicates, 281 unique publications remained. This result equals 91% recall and 18% precision compared to the golden standard. The publications were filtered by our inclusion/exclusion criteria, as shown in Figure 1, and specified in Section 4.1. Borderline articles were discussed in a joint session of the first two authors. Our inclusion/exclusion criteria were validated by having the last two authors compare 10% of the 581 papers retrieved from the primary databases. The comparison resulted in a free-marginal multi-rater kappa of 0.85 [140], which constitutes a substantial inter-rater agreement.

As the next step, we applied the search string to two databases without BibTeX export support. One of them, ACM Digital Library, automatically stemmed the search terms, resulting in more than 1000 search results. The inclusion/exclusion criteria were then applied to the total 1241 publications. This step extended our primary studies by 13 publications, after duplicate removal, and application of inclusion/exclusion criteria, 10 identified in ACM Digital Library and 3 from SciVerse.

As the last step of our publication selection phase, we again conducted exploratory searching. Based on our new understanding of the domain, we scanned the top publication fora and the most published scholars for missed publications. As a last complement, we searched for publications using Google Scholar. In total,

this last phase identified 8 additional publications. Thus, the systematic database search generated 89% of the total number of primary publications, which is in accordance with expectations from the validation of the search string.

As a final validation step, we visualized the selection of the 70 primary publications using REVIS, a tool developed to support SLRs based on visual text mining [74]. REVIS takes a set of primary publications in an extended BibTeX format and, as presented in Figure 4, visualizes the set as a document map (a), edge bundles (b), and a citation network for the document set (c). While REVIS was developed to support the entire SLR process, we solely used the tool as a means to visually validate our selection of publications.

In Figure 4, every node represents a publication, and a black outline distinguishes primary publications (in c), not only primary publications are visualized). In a), the document map, similarity of the language used in title and abstract is presented, calculated using the VSM and cosine similarities. In the clustering, only absolute distances between publications carry a meaning. The arrows point out Antoniol et al.'s publication from 2002 [7], the most cited publication on IR-based trace recovery. The closest publications in a) are also authored by Antoniol et al. [6, 8]. An analysis of a) showed that publications sharing many co-authors tend to congregate. As an example, all primary publications authored by De Lucia et al. [51–54, 56–59], Capobianco et al. [29, 30], and Oliveto et al. [132] are found within the rectangle. No single outlier stands out, indicating that none of the primary publications uses a very different language.

In b), the internal reference structure of the primary studies is shown, displayed by edges connecting primary publications in the outer circle. Analyzing the citations between the primary publications shows one outlier, just below the arrow. The publication by Park *et al.* [134], describing work conducted concurrently with Antoniol *et al.* [7], has not been cited by any primary publications. This questioned the inclusion of the work by Park et al., but as it meets our inclusion/exclusion criteria described in Section 4.1, we decided to keep it.

Finally, in c), the total citation network of the primary studies is presented. Regarding common citations in total, again Park *et al.* [134] is an outlier, shown as I in c). The two other salient data points, II and III, are both authored by Natt och Dag *et al.* [128, 130]. However, according to our inclusion/exclusion criteria, there is no doubt that they should be among the primary publications. Thus, in December 2011, we concluded the set of 70 primary publications.

However, as IR-based trace recovery is an active research field, several new studies were published while this publication was in submission. To catch up with the latest research, we re-executed the search string in the databases listed in Table 4 in June 2012, to catch up with publications from the second half of 2011. This step resulted in 9 additional publications, increasing the number of primary publications to 79. In the rest of this paper, we refer to the original 70 publications as the "core primary publications", and the 79 publications as just the "primary publications".

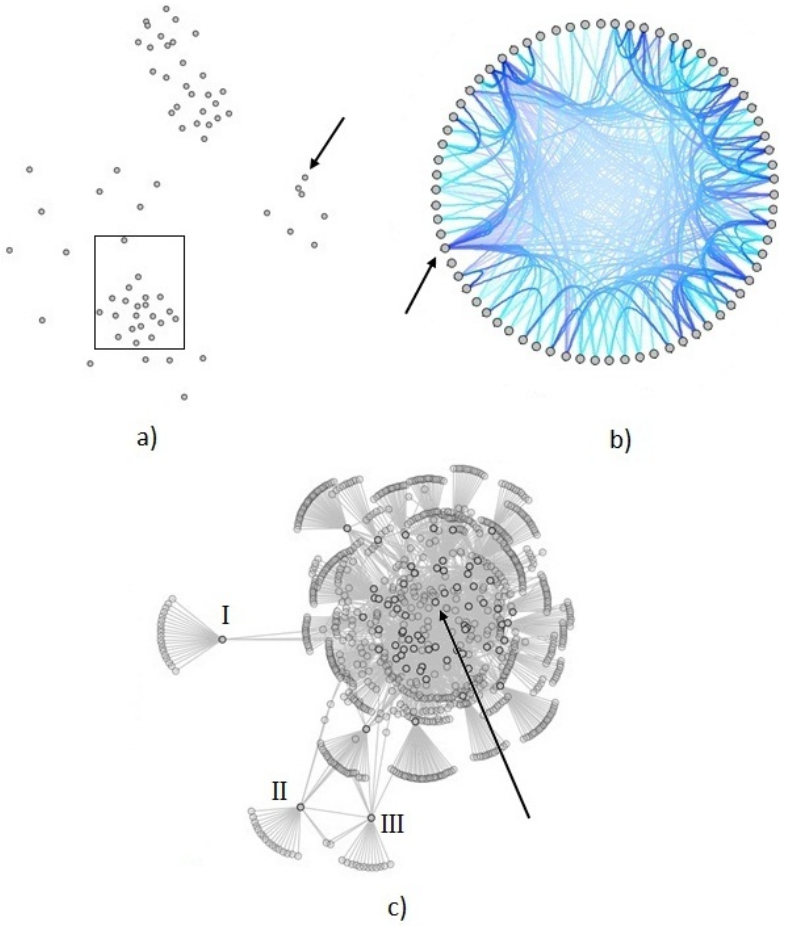


Figure 4: Visualization of core primary publications. a) document map, shows similarities in language among the core primary publications. b) edge bundle, displays citations among the core primary publications. c) citation network, shows shared citations among the core primary publications.

4.3 Data extraction and mapping

During the stage of the study, data was extracted from the primary publications according to the pre-defined extraction form of the review protocol. We extracted general information (title, authors, affiliation, publication forum, citations), details about the applied IR approach (IR model applied, selection and weighting of features, enhancements) and information about the empirical evaluation (types of artifacts linked, size and origin of dataset, research methodology, context of IR evaluation, results of evaluation).

The extraction process was validated by the second and third authors, working on a 30% sample of the core primary publications. Half the sample, 15% of the core primary publications, was used to validate extraction of IR details. The other half was used by the other author to validate empirical details. As expected, the validation process showed that the data extraction activity, and the qualitative analysis inherent in that work, inevitably leads to some deviating interpretations. Classifying according to the four levels of IR contexts, which was validated for the entire 30% sample, showed the least consensus. This divergence, and other minor discrepancies detected, were discussed until an agreement was found and followed for the rest of the primary publications. Regarding the IR contexts in particular, we adopted an inclusive strategy, typically selecting the higher levels for borderline publications.

Regarding results from studies dominated by evaluations in the retrieval and seeking contexts, as described in Section 3.3, we extracted information in two separate ways. For publications comparing multiple underlying IR models, we extracted the conclusions of the original authors, a process that in some cases required qualitative analysis. Also, while the authors of the primary publications based their conclusions on different types of evidence, we treated all conclusions as equally strong.

Also, we extracted P-R values from the primary publications based on our discussions in Section 3.4. However, as highlighted by Banko and Brill in the empirical NLP community [15], the effect of large number of evaluations conducted on small datasets is an open question. One has to wonder what conclusions drawn on small datasets may carry over to datasets of industrial size. In trace recovery research, this is particularly notable regarding context-dependent enhancement strategies, as they risk leading to overtrained or over-engineered solutions that are not generalizable. To mitigate this possible bias, in line with our interest in general solutions that are easily deployable, we extracted P-R values from basic trace recovery methods rather than enhancements. Instead, the various enhancements are reported in Section 5.1.

We agree with the opinion of Spärck Jones *et al.*, pioneers of standardized test collections for IR evaluations, that the widely reported precision at standard recall levels is opaque [158]. The figures obscure the actual numbers of retrieved documents needed to get beyond low recall. Still, we follow the convention established

at TREC and report both precision at fixed levels of recall as well as precision at specific cut-off levels. In line with what was reported for the ad hoc retrieval task at TREC, we report precision at ten recall levels from 0.1 to 1 (referred to as PR@Fix). However, while TREC established the cut-off levels 5, 10, 15, 20, 30 and 100 [98], evaluations on IR-based trace recovery have typically not been reported at such a level of detail. As a consequence, we report P-R values from only the cut-off levels 5, 10 and 100 (referred to as PR@N), as they are the most frequently reported in the primary publications.

Furthermore, as discussed in Section 3.4, neither PR@Fix nor PR@N cover all P-R reporting styles in the primary publications. Thus, we also extracted P-R values beyond standard TREC practice. We extracted P-R values corresponding to a set of candidate trace links with a cosine similarity $\geq$ 0.7 (referred to as PR@Sim0.7). Finally, to ensure that all primary publications reporting P-R values contributed to our synthesis, we also report from an inclusive aggregation of miscellaneous P-R values (referred to as PR@Tot).

While we extracted P-R values from tables whenever possible, a majority were extracted from figures presenting P-R curves. As both the information density and resolution of such figures vary, so did the errors involved in the manual measuring process. Consequently, the extracted data should be interpreted as a general trend rather than exact results.

4.4 Threats to validity

Threats to the validity of the mapping study are analyzed with respect to construct validity, reliability, internal validity and external validity [172]. Particularly, we report deviations from the SLR guidelines [103].

Construct validity concerns the relation between measures used in the study and the theories in which the research questions are grounded. In this study, this concerns the identification of papers, which is inherently qualitative and dependent on the coherence of the terminology in the field. To mitigate this threat, we took the following actions. The search string we used was validated using a golden set of publications, and we executed it in six different publication databases. Furthermore, our subsequent exploratory search further improved our publication coverage. A single researcher applied the inclusion/exclusion criteria, although, as a validation proposed by Kitchenham and Charters [103], another researcher justified 10% of the search results from the primary databases. There is a risk that the specific terms of the search string related to 'activity' (e.g., "requirements tracing") and 'objects' cause a bias toward both requirements research and publications with technical focus. However, the golden set of publications was established by a broad scanning of related work, using both searching and browsing, and was not restricted to specific search terms. Finally, as this work was conducted both as an SM and an SLR, the quality assessment was expressed as a RQ in its own (RQ3). As such, quality was assessed further than by the inclusion/exclusion

criteria. Furthermore, applicable to RQ4, quality differences were accounted for by considering single publications reporting multiple studies as multiple units of empirical evidence.

An important threat to *reliability* concerns whether other researchers would come to the same conclusions based on the publications we selected. The major threat is the extraction of data, as mainly qualitative synthesis was applied, a method that involves interpretation. A single researcher extracted data from the primary publications, and the other two researchers reviewed the process, as suggested by Brereton *et al.* [27]. As a validation, both the reviewers individually repeated the data extraction on a 15% sample of the core primary publications. Another reliability threat is that we present qualitative results with quantitative figures. Thus, the conclusions we draw might depend on the data we decided to visualize; however, the primary studies are publicly available, allowing others to validate our conclusions. Furthermore, as our study contains no formal meta-analysis, no sensitivity analysis was conducted, neither was publication bias explored explicitly.

Internal validity concerns confounding factors that can affect the causal relationship between the treatment and the outcome, especially relevant to RQ4. There is a threat that the reporting style in the primary publications has a bigger impact on our conclusions than the actual output from tools. Consequently, there is a risk that we failed to include results due to differences in both experimental setups and level of detail in reports. Also regarding RQ4, we aggregate evidence from previous comparisons made in different ways, some report statistical analysis while others discuss results in more general terms. We do not weight the contributions of the individual studies based on this. Moreover, while we attempted to distinguish between query-based and matrix-based evaluations in the synthesis, different contexts (e.g., domain, work task, artifact types, language of the artifacts) were all included in the synthesis of P-R values.

External validity refers to generalization from this study. As we do not claim that our results apply to other applications of IR in software engineering, this is a minor threat. On the other hand, due to the comprehensive nature of our study, we extrapolate our conclusions on RQ4 to all studies on IR-based trace recovery published until December 2011, including studies that did not report a sufficient amount of details.

5 Results

Following the method defined in Section 4.2, we identified 79 primary publications. Most of the publications were published in conferences or workshops (67 of 79, 85%), while twelve (15%) were published in scientific journals. Table 5 presents the top publication channels for IR-based trace recovery, showing that it spans several research topics. Figure 5 depicts the number of primary publications

Publication forum	#Publications
International Requirements Engineering Conference	9
International Conference on Automated Software Engineering	7
International Conference on Program Comprehension	6
International Workshop on Traceability in Emerging Forms of Software Engineering	6
Working Conference on Reverse Engineering	5
Empirical Software Engineering	4
International Conference on Software Engineering	4
International Conference on Software Maintenance	4
Other publication fora (two or fewer publications)	34

Table 5: Top publication channels for IR-based trace recovery.

per year, starting from Antoniol et al.'s pioneering work from 1999. Almost 150 authors have contributed to the 79 primary publications, on average writing 2.2 of the articles. The top five authors have on average authored 14 of the primary publications, and are in total included as authors in 53% of the articles. Thus, a wide variety of researchers have been involved in IR-based trace recovery, but there is a group of a few well-published authors. More details and statistics about the primary publications are available in Appendix 6.

Several publications report empirical results from multiple evaluations. Consequently, our mapping includes 132 unique empirical contributions, i.e., the mapping comprises results from 132 unique combinations of an applied IR model and its corresponding evaluation on a dataset. As described in Section 4.1, we denote

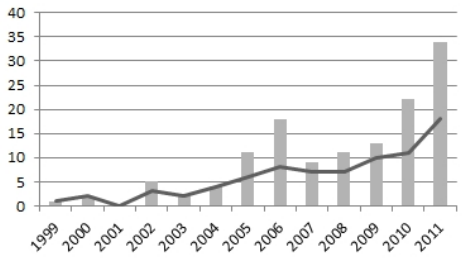


Figure 5: IR-based trace recovery publication trend. The curve shows the number of publications, while the bars display empirical studies in these publications.

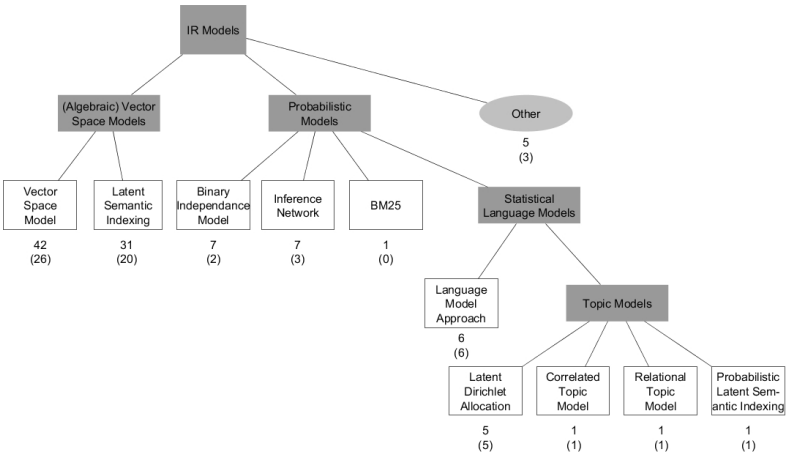


Figure 6: Taxonomy of IR models in trace recovery. The numbers show in how many of the primary publications a specific model has been applied, the numbers in parentheses show IR models applied since 2008.

such a unit of empirical evidence a 'study', to distinguish from 'publications'.

5.1 IR models applied to trace recovery (RQ1)

In Figure 6, reported studies in the primary publications are mapped according to the (one or several) IR models applied, as defined in Section 2. The most frequently reported IR models are the algebraic models, VSM and LSI. Various probabilistic models have been applied in 29 of the 113 evaluations, including 14 applications of statistical LMs. Five of the applied approaches identified do not fit in the taxonomy; examples include utilizing swarm techniques [160] and B-splines [30]. As shown in Figure 6, VSM has been the most applied model 2008-2011, however repeatedly as a benchmark to compare new IR models against. An apparent trend is that trace recovery based on LMs has received an increasing research interest during the last years.

Only 46 (71%) of the 65 primary publications with technical foci report which preprocessing operations were applied to NL text. Also, in several publications one might suspect that the complete preprocessing was not reported. As a result, a reliable report of feature selection for IR-based trace recovery is not possible. Furthermore, several papers do not report any differences regarding preprocessing of NL text and source code. Among the publications reporting preprocessing, 29 report conducting stop word removal and stemming, making it the most common combination. The remaining publications report other combinations of stop word removal, stemming and ID splitting. Also, two publications report applying

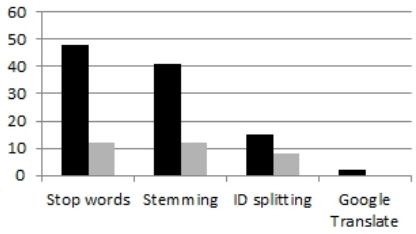


Figure 7: Preprocessing operations used in IR-based trace recovery. The figure shows the number of times a specific operation has been reported in the primary publications. Black bars refer to preprocessing of NL text, gray bars show preprocessing of text extracted from source code.

Google Translate as a preprocessing step to translate NL text to English [91, 110]. Figure 7 presents in how many primary publications different preprocessing steps are explicitly mentioned, both for NL text and source code.

Regarding NL text, most primary publications select all terms that remain after preprocessing as features. However, two publications select only nouns and verbs [176, 177], and one selects only nouns [30]. Also, Capobianco *et al.* have explicitly explored the semantic role of nouns [29]. For the purposes of the mapping of primary publications dealing with source code, a majority unfortunately does not clearly report about the feature selection (i.e., selecting which subset of terms to extract to represent the artifact). Seven publications report that only IDs were selected, while four publications selected both IDs and comments. Three other publications report more advanced feature selection, including function arguments, return types and commit comments [1, 3, 28].

Among the primary publications, the weighting scheme applied to selected features is reported in 58 articles. Although arguably more tangible for algebraic retrieval models, feature weighting is also important in probabilistic retrieval. Moreover, most weighing schemes are actually families of configuration variants [149], but since this level of detail often is omitted in publications on IR-based trace recovery [131], we were not able to investigate this further. Figure 8 shows how many times, in the primary publications, various types of feature weighting schemes have been applied. Furthermore, one publication reports upweighting of verbs in the TFIDF weighting scheme, motivated by verbs' nature of describing the functionality of software [117].

Several strategies to improve the performance of IR-based trace recovery tools are proposed, as presented in Figure 9. The figure shows how many times different enhancement strategies have been applied in the primary publications. Most enhancements aim at improving the precision and recall of the tool output, however also a (computation) performance enhancement is reported [97]. The most fre-

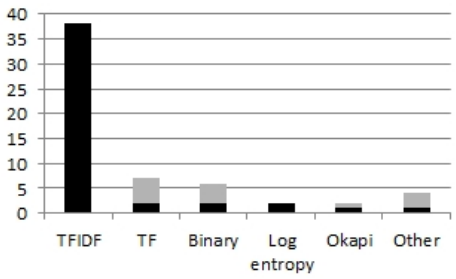


Figure 8: Feature weighting schemes in IR-based trace recovery. Bars depict how many times a specific weighting scheme has been reported in the primary publications. Black color shows reported weighting in publications applying algrebraic IR models.

quently applied enhancement strategy is relevance feedback, giving the human a chance to judge partial search results, followed by re-executing an improved search query. The following most frequently applied strategies are applying a thesaurus to deal with synonyms, clustering (organizing/ranking results depending on for instance document structure), phrasing (going beyond the bag of word model by considering sequences of words). Other enhancement strategies repeatedly applied include: up-weighting terms considered important by applying a project glossary, machine learning approaches to improve results based on for example the existing trace link structure, and combining the results from different retrieval models in voting systems. Yet another set of enhancements have only been proposed in single primary publications, such as query expansion, analyses of call graphs, regular expressions, and smoothing filters.

5.2 Types of software artifacts linked (RQ2)

Figure 10 maps onto the classical software development V-model the various software artifact types that have been used in IR-based trace recovery evaluations. Requirements, the left part of the model, include all artifacts that specify expectations on a system, e.g., market requirements, system requirements, functional requirements, use cases, and design specifications. The distinction between these are not always possible to derive from the publications, and hence we have grouped them together under the broad label ‘requirements’. The right part of the model represents all artifacts related to verification activities, e.g., test case descriptions and test scripts. Source code artifacts constitute the bottom part of the model. Note however, that our inclusion/exclusion criteria, excluding duplication analyses and studies where not either source or target artifacts are dominated by NL text, results in fewer links between requirements-requirements, code-code, code-test, test-test

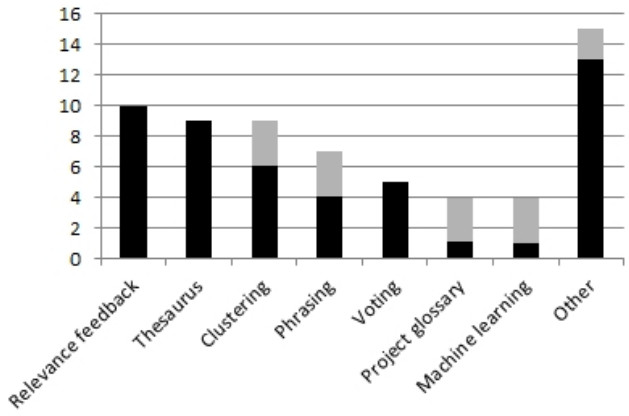


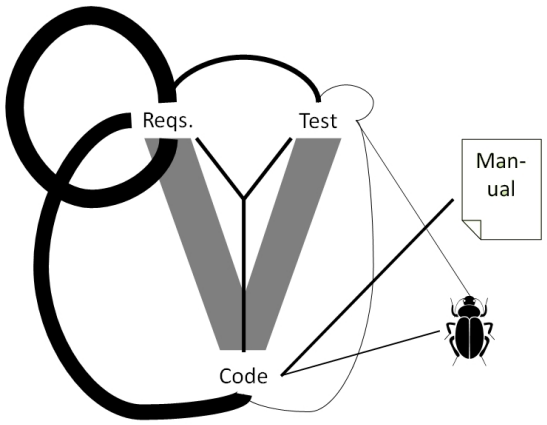
Figure 9: Enhancement strategies in IR-based trace recovery. Bars depict how many times a specific strategy has been reported in the primary publications. Black color represents enhancements reported in publications using algebraic IR models.

and defect-defect than would have been the case if we had studied the entire field of IR applications within software engineering.

The most common type of links that has been studied was found to be between requirements (37 evaluations), either of the same type or of different levels of abstraction. The second most commonly studied artifact linking is between requirements and source code (32 evaluations). Then, in decreasing order, mixed links in an information space of requirements, source code and tests (10 evaluations), links between requirements and tests (9 evaluations) and links between source code and manuals (6 evaluations). Less frequently studied trace links include links between source code and defects/change requests and links between tests. In three primary publications, the types of artifacts traced are unclear, either not specified at all or merely described as 'documents'.

5.3 Strength of evidence (RQ3)

An overview of the datasets used for evaluations in the primary publications is shown in Figure 1. In total we identified 132 evaluations; in 42 (32%) cases proprietary artifacts, either originating from development projects in private companies or the US agency NASA, were studied. Nineteen (14%) evaluations using artifacts collected from open source projects have been published and 65 (49%) employing artifacts originating from a university environment. Among the datasets from university environments, 34 consist of artifacts developed by students. In six primary publications, the origin of the artifacts is mixed or unclear. Figure 1 also depicts the sizes of the datasets used in the evaluations, wrt. the number of arti-



Reqs.-Reqs.	37
Code-Reqs.	32
Reqs.-Test	10
Reqs.-Code-Test	9
Code-Manual	6
Code-Defect	3
Code-Test	2
Test-Test	1
Test-Defect	1

Figure 10: Types of links recovered in IR-based trace recovery. The table shows the number of times a specific type of link is the recovery target in the primary publications, also represented by the weight of the edges in the figure.

facts. The majority of the evaluations in the primary publications were conducted using an information space of less than 500 artifacts. In 38 of the evaluations, less than 100 artifacts were used as input. The primary publications with the by far highest number of artifacts, evaluated links between 3.779 business requirements and 8.334 market requirements at Baan [128] (now owned by Infor Global Solutions), and trace links between 9 defect reports and 13380 test cases at Research in Motion [100].

Table 6 presents the six datasets that have been most frequently used in evaluations of IR-based trace recovery, sorted by the number of primary studies in which they were used. CM-1, MODIS, and EasyClinic are publicly available from the CoEST web page[4]. Note that most publicly available datasets except EasyClinic are *bipartite*, i.e., the dataset contains only links between two disjunct subsets of artifacts.

All primary publications report some form of empirical evaluations, a majority (80%) conducting "studies of methods" [88]. Fourteen publications (18%) report results regarding the human analyst, two primary publications study both methods and human analysts. Figure 12 shows the primary publications mapped to the four levels of the context taxonomy described in Section 3.3. Note that a number of publications cover more than one environment, due to either mixed artifacts or multiple studies. Also, two publications did not report the environment, and could not be mapped. A majority of the publications (50), exclusively conducted evaluations taking place in the innermost retrieval context, the so-called "cave of IR evaluation" [92]. As mentioned in Section 2, evaluations in the cave display an

[4]coest.org

#	Dataset	Artifacts	Links	Origin	Development characteristics	Size[a]
17	CM-1	Requirements specifying system requirements and detailed design	Bipartite dataset, many-to-many links	NASA	Embedded software development in governmental agency	455
16	EasyClinic	Use cases, sequence diagrams, source code, test case descriptions. Language: Italian.	Many-to-many links	Univ. of Sannio	Student project	150
8	MODIS	Requirements specifying system requirements and detailed design	Bipartite dataset, many-to-many links.	NASA	Embedded software development in governmental agency	68
7	Ice-Breaker System (IBS)	Functional requirements and source code	Not publicly available in full detail	[142]	Textbook on requirements engineering	185
6	LEDA	Source code and user documentation	Bipartite dataset, many-to-one links	Max Planck Inst. for Informatics Saarbrücken	Scientific computing	296
5	Event-Based Traceability (EBT)	Functional requirements and source code	Not publicly available	DePaul Univ.	Tool from research project	138

[a] Size is presented as the total number of artifacts.

Table 6: Summary of the datasets most frequently used for evaluations.

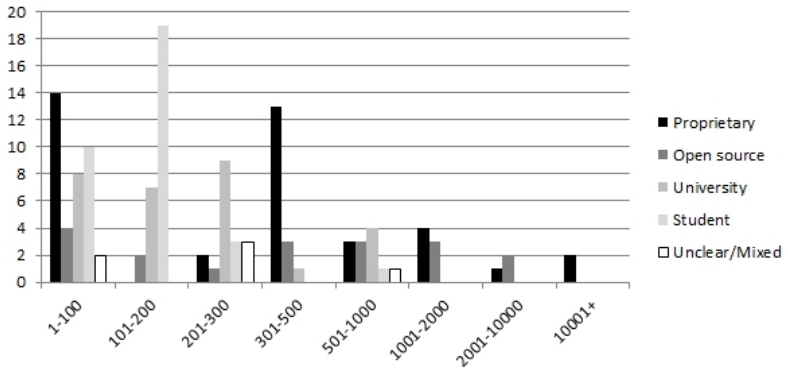


Figure 11: Datasets used in studies on IR-based trace recovery. Bars show number and origin of artifacts.

inconsistent use of terminology. Nineteen (38%) of the primary publications refer to their evaluations in the retrieval context as experiments, 22 (44%) call them case studies, and in nine (18%) publications they are merely referred to as studies.

Since secondary measures were applied, fourteen publications (18%) are considered to have been conducted in the seeking context. Eleven primary publications conducted evaluations in the work context, mostly through controlled experiments with student subjects. Only three evaluations are reported in the outermost context of IR evaluation, the project context, i.e., evaluating the usefulness of trace recovery in an actual end user environment. Among these, only a single publication reports an evaluation from a non-student development project [110].

5.4 Patterns regarding output accuracy (RQ4)

Meta-analysis was not possible due to inhomogeneous experimental setups and presentation. Therefore, we applied vote counting analysis. Among the primary publications, 25 compare the output accuracy of trace recovery when applying different IR models. Figure 13 depicts the outcomes of the comparisons, based on the original authors' conclusions. An edge represents a comparison of two implemented IR models on a dataset, thus a single publication can introduce several arrows. The direction of an edge points at the IR model that produced the most accurate output, i.e., an arrow points at the better model. Accordingly, an undirected edge (dotted) shows inconclusive comparisons. Finally, an increased edge weight depicts multiple comparisons between the IR models, also presented as a label on the edge.

VSM and LSI are the two IR models that have been most frequently evaluated in comparing studies. Among those, and among comparing studies in general,

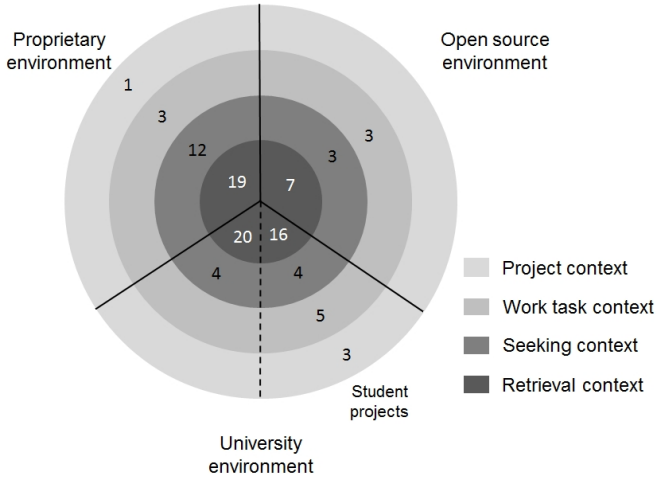
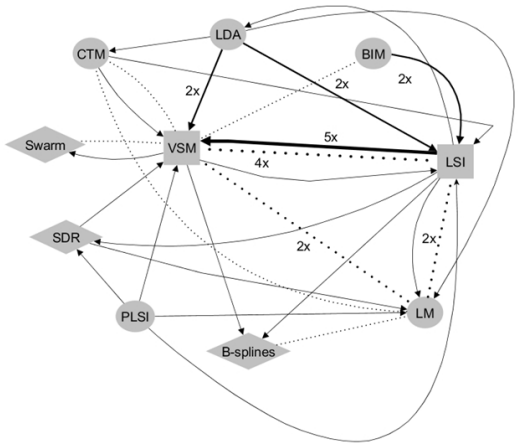


Figure 12: Contexts of evaluations of IR-based trace recovery, along with study environments. Numbers show the number of primary publications that target each combination.

VSM has presented the best results. On the other hand, implementing LMs and measuring similarities using JS divergence has been concluded as a better technique in four studies and has never underperformed any other models. As it has been compared to VSM in three publications, it appears to perform trace recovery with a similar accuracy [1, 30, 132]. Also conducting retrieval based on B-splines and swarm techniques has not been reported to perform worse than other models, but has only been explored in two primary publications [30, 160]. Notably, one of the commonly applied IR models, PIN, has not been explicitly studied in comparison to other IR models within the context of trace recovery.

We also present P-R values extracted from the primary publications as shown in Figures 14 and 15. In the upper right corners of figures, constituting the ideal output accuracy of an IR-based trace recovery tool, we show intervals representing 'Excellent', 'Good', and 'Acceptable' as proposed by Huffman Hayes *et al.* [88]. As discussed in Section 3.4, it is unclear in several primary publications whether a query-based or matrix-based evaluation style has been used. However, it is apparent that a majority of the P-R values in PR@5/10/100 originate from query-based evaluations, and that the P-R values in PR@Sim0.7/Fix/Tot are dominated by matrix-based evaluations.

Figure 14 show P-R footprints from trace recovery evaluations with constant cutpoints at 5, 10, and 100 candidate trace links respectively. Evaluations on five datasets (Leda, CM-1, Gedit, Firefox, and ArgoUML) are marked with separate symbols, as shown in the legend. Especially for PR@5 and PR@10, the primary publications contain several empirical results from trace recovery on these



IR model	Better	Worse	Diff
VSM	10	3	+7
LM	4	0	+4
B-splines	2	0	+2
Swarms	1	0	+1
SDR	2	2	0
LSI	8	9	-1
CTM	1	2	-1
BIM	0	2	-2
PLSI	0	4	-4
LDA	1	6	-5

Figure 13: Empirical comparisons of IR models in trace recovery. Squares show algebraic models, circles represent probabilistic models, diamonds show other models.

datasets. The P-R values in PR@5, PR@10 and PR@100 represent evaluations using: LDA (24 results), LSI (19 results), BM25 (18 results), VSM (14 results), LM (12 results), BIM (7 results), PLSI (6 results), and SDR (6 results).

No clear pattern related to the IR models can be observed in Figure 14. Instead, *different* implementations performed *similarly* when applied to the *same* datasets. This is particularly evident for evaluations on LEDA, for which we could extract several P-R values (shown as squares in Figure 14). In the footprint PR@5, nine P-R values from four different research groups implementing VSM [7], BIM [7], and LSI [97, 121, 167] cluster in the lower right corner. Also, the footprint PR@10 shows five results from evaluations on LEDA, clustered in the very right corner, corresponding to P-R values from three different research groups implementing VSM [7], BIM [7], and LSI [97, 121]. No results on LEDA have been reported at the constant cut-off 100. In line with the results on LEDA, PR@5/10/100 show clustered P-R values from trace recovery using different configurations of BM25 on the Firefox dataset (diamonds) and Gedit (triangles). Similar results can be seen regarding evaluations on CM-1 (circles) in PR@5 and ArgoUML (pluses) in PR@100. However, results on CM-1 in PR@10/100 and ArgoUML in PR@5/10 are less clear as they display lower degrees of clustering.

Few P-R values originating from query-based evaluations are within the three goodness zones. Seven P-R values, corresponding to evaluations on LEDA (4 results), Firefox (2 results), and MODIS (1 result) are within the 'acceptable' zone in PR@5. On the other hand, no P-R values within any goodness zones have been reported in the primary publications neither in PR@10 nor PR@100. In general, this is due to unacceptable precision values. In Figure 14, a number of P-

R values display precision values above 0.4, however they represent matrix-based evaluation of trace recovery in the EasyClinic and eTour datasets.

In studies where constant cut-points have not been used, P-R values in all goodness zones have been reported. In the footprint PR@Sim0.7 in Figure 15, showing P-R values corresponding to candidate trace links with a cosine similarity of ≥ 0.7, P-R values are located in the entire P-R space. Four 'good' values are reported from a publication recovering trace links in the EasyClinic dataset [53]. In PR@Fix, the expected precision-recall tradeoff is evident as shown by the trendline. Several primary publications report both 'acceptable' and 'good' P-R values. Six P-R values are even reported within the 'excellent' zone, all originating from evaluations of trace recovery based on LSI. However, all six evaluations were conducted on datasets containing around 150 artifacts, CoffeeMaker (5 results) and EasyClinic (1 result). In PR@Tot, showing 1.076 P-R values, 19 (1.8%) are in the 'excellent' zone. Apart from the six P-R values that are also present in PR@Fix, additional 'excellent' results have been reported on EasyClinic based on LSI (6 results) and VSM (1 result). Also, P-R values in the 'excellent' zone have been reported from evaluations of VSM-based recovery of trace links between documentation and source code on JDK1.5 (2 results) [34], trace recovery in the MODIS dataset (1 result) [162] and, also implementing VSM, from recovered trace links in an undisclosed dataset (2 results) [135].

In total, we extracted 270 P-R values (25.2%) within the 'acceptable' zone, 129 P-R values (12.0%) in the 'good' zone, and 19 P-R values (1.8%) in the 'excellent' zone. The average (balanced) F-measure for the P-R values in PR@Tot is 0.31 with a standard deviation of 0.07. The F-measure of the lowest acceptable P-R value is lower, 0.24 (corresponding to recall=0.6, precision=0.2), which reflects the difficulty in achieving reasonably balanced precision and recall in IR-based trace recovery. Among the 100 P-R values with the highest F-measure in PR@Tot, 69 have been reported when evaluating trace recovery on the EasyClinic dataset, extracted from 9 different publications. However, the other 31 P-R values come from evaluations on 9 other datasets originating from either industrial, open source or academic contexts.

6 Discussion

Our set of 79 primary publications shows that there are more publications on IR-based trace recovery than has previously been acknowledged. Based on our comprehensive study, the rest of this section discusses our research questions. Along with the discussions, we conclude every question with concrete suggestions on how to advance research on IR-based trace recovery. Finally, in Section 6.5, we map our recommendations to the traceability challenges articulated by CoEST [79].

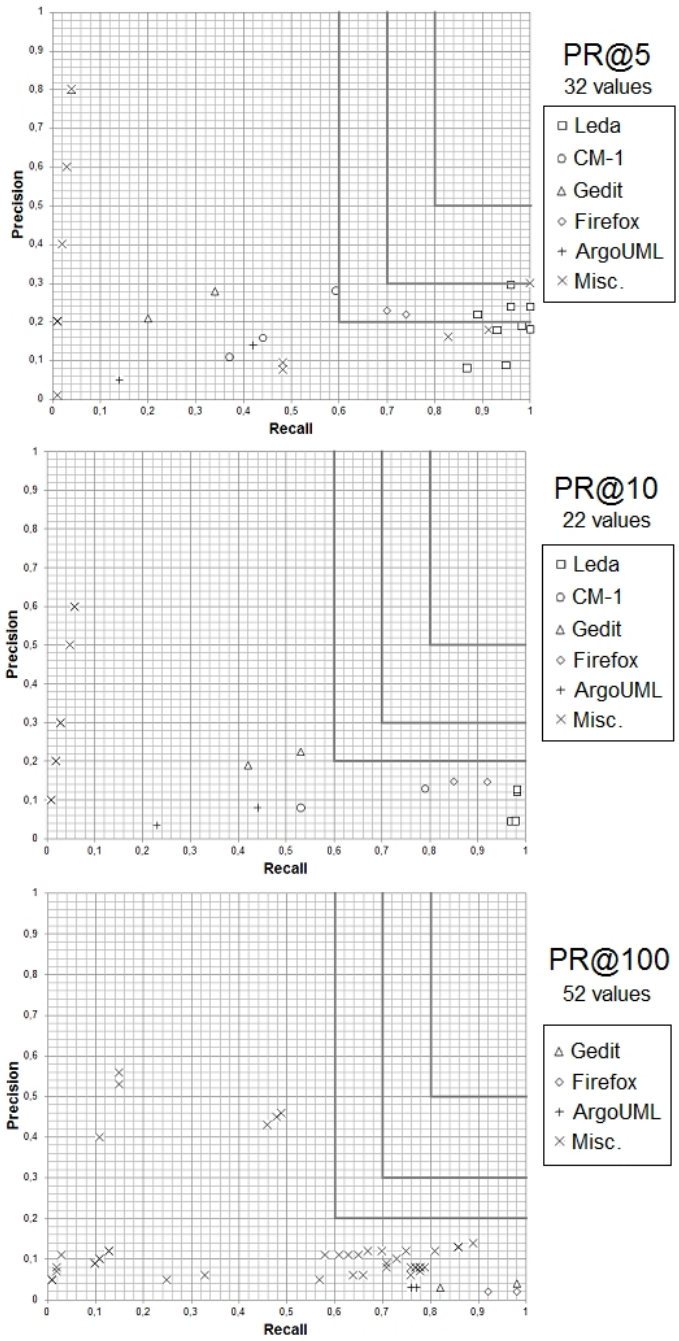


Figure 14: P-R footprints for trace recovery tools. The figures show P-R values at the constant cut-offs PR@5, PR@10 and PR@100.

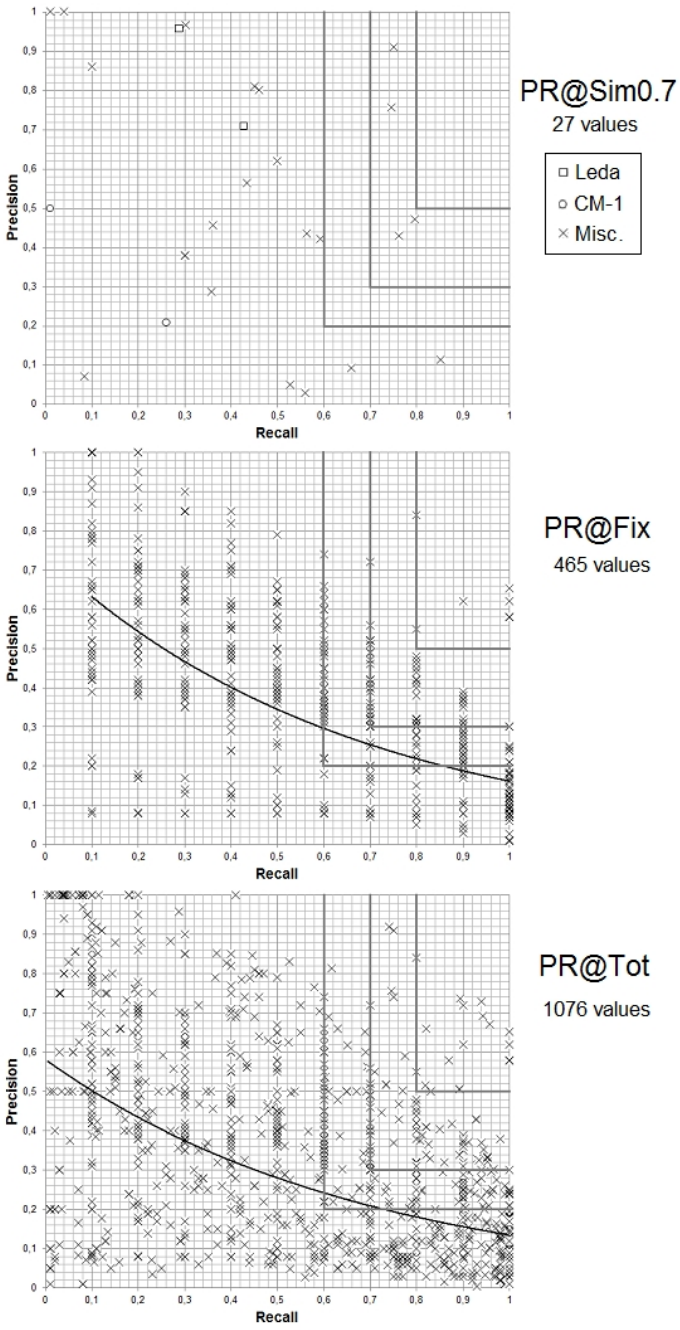


Figure 15: P-R footprints for trace recovery tools. The figures show P-R values representing a cut-off at the cosine similarity 0.7 (PR@Sim0.7), precision at fixed recall levels (PR@Fix), and an aggregation of all collected P-R values (PR@Tot). The figures PR@Fix and PR@Tot also present a P-R curve calculated as an exponential trendline.

6.1 IR models applied to trace recovery (RQ1)

During the last decade, a wide variety of IR models have been applied to recover trace links between artifacts. Our study shows that the most frequently applied models have been algebraic, i.e., Salton's classic VSM from the 60s [150] and LSI, the enhancement developed by Deerswester in the 90s [60]. Also, we show that VSM has been implemented more frequently than LSI, in contrast to what was reported by Binkley and Lawrie [18]. The interest in algebraic models might have been caused by the straightforwardness of the techniques; they have concrete geometrical interpretations, and are rather easy to understand also for non-IR experts. Moreover, several open source implementations are available. Consequently, the algebraic models are highly applicable to trace recovery studies, and they constitute feasible benchmarks when developing new methods. However, in line with the development in the general IR field [173], LMs [138] have been getting more attention in the last years.

While implementing an IR model, the developers inevitably have to make a variety of design decisions. Consequently, this applies also to IR-based trace recovery tools. As a result, tools implementing the same IR model can produce rather different output [23]. Thus, omitting details in the reporting obstructs the possibility to advance the field of trace recovery through secondary studies and evidence-based software engineering techniques [96]. Unfortunately, even fundamental information about the implementation of IR is commonly left out in trace recovery publications. Concrete examples include feature selection and weighting (particularly neglected for publications indexing source code) and the number of dimensions of the LSI subspace. Furthermore, the heterogeneous use of terminology is an unnecessary difficulty in IR-based trace recovery publications. Concerning general traceability terminology, improvements can be expected as Cleland-Huang *et al.* dedicated an entire chapter of their recent book to this issue [39]. However, we hope that Section 1 of this paper is a step toward aligning also the IR terminology in the community.

To support future replications and secondary studies on IR-based trace recovery, we suggest that:

- Studies on IR-based trace recovery should use IR terminology consistently, e.g., as presented in Table 1 and Figure 6, and use general traceability terminology as proposed by Cleland Huang *et al.* [39].
- Authors of articles on IR-based trace recovery should carefully report their implemented IR model, to enable aggregating empirical evidence.
- Technology-oriented experiments on IR-based trace recovery should adhere to rigorous methodologies such as the framework by Huffman Hayes and Dekhtyar [84].

6.2 Types of software artifacts linked (RQ2)

Most published evaluations on IR-based trace recovery aim at establishing trace links between requirements of different kinds, or between requirements and source code. Apparently, the artifacts of the V&V side of the V-model are not as frequently in focus of researchers working on IR-based trace recovery. One can think of several reasons for this unbalance. First, researchers might consider that the structure of the document subspace of the requirement side of the V-model is more important to study. Second, the early public availability of a few datasets containing requirements of various kinds, might have paved the way for a series of studies by various researchers. Third, publicly available artifacts from the open source community might contain more requirements artifacts than V&V artifacts. Nevertheless, research on trace recovery would benefit from studies on a more diverse mix of artifacts. For instance, the gap between requirements artifacts and V&V artifacts is an important industrial challenge [148]. Hence, exploring whether IR-based trace recovery could be a way to align "the two ends of software development" is worth an effort.

Apart from the finding that requirement-centric studies on IR-based trace recovery are over-represented, we found that too few studies go beyond trace recovery in bipartite traceability graphs. Such simplified datasets hardly represent the diverse information landscapes of large-scale software development projects. Exceptions include studies by De Lucia et al., who repeatedly have evaluated IR-based trace recovery among use cases, functional requirements, source code and test cases [47, 51, 53, 56–59], however originating from student projects, which reduces the industrial relevance.

To further advance the research of IR-based trace recovery, we suggest that:

- Studies should be conducted on diverse datasets containing a higher number of artifacts.
- Studies should go beyond bipartite datasets to better represent the heterogeneous information landscape of software engineering.

6.3 Strength of evidence (RQ3)

Most evaluations on IR-based trace recovery were conducted on bipartite datasets containing fewer than 500 artifacts. Obviously, as pointed out by several researchers, any software development project involves much larger information landscapes, that also consist of heterogeneous artifacts. A majority of the evaluations of datasets containing more than 1,000 artifacts were conducted using open source artifacts, an environment in which fewer types of artifacts are typically maintained [28, 151], thus links to or from source code are more likely to be studied. Even though small datasets might be reasonable to study, only two

primary publications report from evaluations containing more than 10,000 artifacts [100, 128]. As a result, the question of whether the state-of-the-art IR-based trace recovery scales to larger document spaces or not, commonly mentioned as future work [54, 78, 90, 107, 118, 167] remains unanswered, is a major threat to external validity.

Regarding the validity of datasets used in evaluations, a majority used artifacts originating from university environments as input. Furthermore, most studies on proprietary artifacts used only the CM-1 or MODIS datasets collected from NASA projects, resulting in their roles as de-facto benchmarks from an industrial context. Clearly, again the external validity of state-of-the-art trace recovery must be questioned. On one hand, benchmarking can be a way to advance IR tool development, as TREC have demonstrated in the general IR research [155], but on the other hand it can also lead the research community to over-engineering tools on specific datasets [23]. The benchmark discussion has been very active in the traceability community the last years [16, 37, 62, 63, 79].

A related problem, in particular for proprietary datasets that cannot be disclosed, is that datasets often are poorly described [24]. In some particular publications, NL artifacts in datasets are only described as 'documents'. Thus, as already discussed related to RQ1 in Section 6.1, inadequate reporting obstructs replications and secondary studies. Moreover, as Figure 14 shows, the choice of datasets in evaluations of IR-based trace recovery can impact the tool output far more than the choice of IR model, in line with results by Ali *et al.* [4].

Most empirical evaluations of IR-based trace recovery were conducted in the innermost of IR contexts, i.e., a clear majority of the research was conducted "in the cave" or just outside [92]. For some datasets, the output accuracy of IR models has been well-studied during the last decade. However, more studies on how humans interact with the tools are required; similar to what has been explored by Huffman Hayes et al. [45, 64, 85, 90] and De Lucia et al. [56–58]. Thus, more evaluations in a work task context or a project context are needed. Regarding the outermost IR context, only one industrial in-vivo evaluation [110] and three evaluations in student projects [52–54] have been reported. Finally, regarding the innermost IR contexts, the discrepancy of methodological terminology should be harmonized in future studies.

To further advance evaluations of IR-based trace recovery, we suggest that:

- The community should continue its struggle to acquire a set of more representative benchmarks.
- Researchers should better characterize the datasets used in evaluations, in particular when they cannot be disclosed for confidentiality reasons.
- An industrial case study, even a small but well-conducted study, should be highly encouraged as an important empirical contribution.

6.4 Patterns regarding output accuracy (RQ4)

We synthesized P-R values from 48 primary publications and concluded that there is no empirical evidence that the extensive research on new IR models for trace recovery has improved the accuracy of the candidate trace links. Hence, our results confirm previous findings by Oliveto *et al.* [132] and Binkley and Lawrie [18], that no IR model is consequently outperforming others. Instead, our results suggest that the classic VSM, developed by Salton *et al.* in the 60s [150], performs better or as good as other models. Our findings are also in line with the claim by Falessi *et al.*, that simple IR techniques are typically the most useful [73]. Thus, as also pointed out by Ali *et al.* [4], we see little value for the traceability community to continue publishing studies that solely hunt improved P-R values "in the cave", without considering other factors that impact trace recovery, e.g., the validity of the dataset and the specific work task the tools are intended to support.

Furthermore, as Cuddeback *et al.* rather controversially highlighted, human subjects vetting entire candidate traceability matrices do not necessarily benefit from more accurate candidate trace links [45]. Instead, Cuddeback *et al.* showed that humans tend to vet the candidate traceability matrix in a way that balances precision and recall. Furthermore, while humans provided with low accuracy candidate traceability matrices improved them significantly, humans vetting highly accurate candidate traceability matrices often decreased their accuracy. These findings have also been statistically confirmed by Dekhtyar *et al.* [61], in a study with 84 subjects. While these findings concern matrix-based evaluations, Borg and Pfahl explored this phenomenon preliminary in a query-based evaluation environment [22]. In a pilot experiment on impact analysis, subjects were provided with lists of candidate trace links, i.e., one ranked search list per artifact to trace, representing different accuracy levels. While the results were inconclusive, there were indications that subjects benefited from more accurate tool output. However, also in this experiment, humans tended to complete the task in a way that balanced the precision and recall of the final set of trace links. More human-oriented research is needed, including visualization of trace links as has been initially compassed by Marcus *et al.* [123] and Chen [34].

Regarding matrix-based evaluations of IR-based trace recovery, the aggregation of precision at fixed levels clearly displays the expected trade-off between the two measures. Also, when comparing to the quality levels defined by Huffman Hayes *et al.* [88], the challenge of reaching 'acceptable' precision and 'acceptable' recall is evident, as it is only achieved in about a quarter of the reported P-R values. While the appropriateness of the proposed quality levels (originally presented by Huffman Hayes *et al.* as an attempt to "draw a line in the sand"), cannot be validated without user studies, they constitute a starting point for the synthesis of empirical results. Some published results are 'acceptable', a few are even 'good' or 'excellent', while a majority of the results are 'unacceptable'. However, more work similar to what Cuddeback *et al.* [45] and Dekhtyar *et al.* [61] have presented

is required to validate the quality levels.

Our findings also confirm the difficulty to determine which set of candidate trace links to present to the user of an IR-based trace recovery tool, i.e, deciding which cut-off strategy is the most feasible for the specific context. The accuracy of a set of trace links is unknown at recovery time of the IR tool, at least unless it can be compared to a set of pre-established trace links. Thus, as only a quarter of the reported P-R values are 'acceptable', this supports the suggestion by De Lucia *et al.*, that an engineer should work incrementally with output from IR-based trace recovery tools [57]. As described by De Lucia *et al.*, the engineer can then iteratively balance on the P-R trade-off, in a manner that is suitable for the work task at hand. However, while we argue that the fact that a user can achieve more efficient results with specific ways of using a tool, it does not mean that studies should report non-conventional P-R values in place of PR@Fix and PR@N. This is especially true regarding PR@N, as publications on IR-based trace recovery often hide the number of candidate trace links required to reach a certain level of recall, i.e., only PR@Fix is reported. Thus, as argued by Spärck Jones *et al.* [158], researchers should make an effort to also report PR@N to display the amount of information a user would have to process. Obviously, in the case of a matrix-based evaluation in a large document space a very high N might be required to recover enough trace links, but still it is meaningful to report this piece of information. Concerning query-based evaluations of IR-based trace recovery, the proposed quality levels do not seem to be appropriate, thus PR@Fix and PR@N should instead be evaluated using secondary measures such as MAP and DCG.

As discussed in Section 6.3, several studies have reported that IR-based trace recovery tools support humans when performing tasks involving tracing. Thus, as a majority of the P-R values reported by state-of-the-art IR-based trace recovery tools does not reach 'acceptable' accuracy levels, this suggests that even supporting an engineer with rather inaccurate tool support is better than working manually with tracing tasks. This seems reasonable also in the light of work by Cuddeback *et al.* [45] and Dekhtyar *et al.* [61], in which they show that human subjects provided poor starting points for tracing improve them significantly. Consequently, as much state-of-the-practice tracing work is done in environments with rather limited tool support [21, 88], providing an engineer with state-of-the-art candidate trace links has a potential to increase the traceability in an industrial project, as well as the related concept *findability*, defined as "the degree to which a system or environment supports navigation and retrieval" [127].

To strengthen the validity of evaluations of IR-based trace recovery, we suggest that:

- Results should be carefully reported using PR@Fix and PR@N, complemented by secondary measures.

Research theme	Goal to reach by 2035
Purposed traceability	to define and instrument prototypical traceability profiles and patterns
Cost-effective traceability	to perform systematic quality assessment and assurance of the traceability
Configurable traceability	to provide for levels of abstraction and granularity in traceability techniques, methods and tools, facilitated by improved trace visualisations, to handle very large datasets and the longevity of these data
Trusted traceability	to develop cost-benefit models for analysing stakeholder requirements for traceability and associated solution options at a fine-grained level of detail
Scalable traceability	to use dynamic, heterogeneous and semantically rich traceability information models to guide the definition and provision of traceability
Portable traceability	to agree upon universal policies, standards, and a unified representation or language for expressing traceability concepts
Valued traceability	to raise awareness of the value of traceability, to gain buy-in to education and training, and to get commitment to implementation
Ubiquitous traceability	to provide automation such that traceability is encompassed within broader software and systems engineering processes, and is integral to all tool support

Table 7: Traceability research themes defined by CoEST [79]. Ubiquitous traceability is referred to as "the grand challenge of traceability", since it requires significant progress in the other research themes.

- It should be made clear whether a query-based or matrix-based evaluation style has been used, especially when reporting P-R values.
- Focus on tool enhancements "in the cave" should be shifted towards evaluations in the work task or project context.

6.5 In the light of the CoEST research agenda

Gotel *et al.* recently published a framework of challenges in traceability research [79], a CoEST community effort based on a draft from 2006 [40]. The intention of the framework is to provide a structure to direct future research on traceability. CoEST defines eight research themes, addressing challenges that are envisioned as solved in 2035, as presented in Table 7. Our work mainly contributes to three of the research themes, *purposed traceability*, *trusted traceability*, and *scalable traceability*. Below, we discuss the three research themes in relation to IR-based trace recovery, based on our empirical findings.

The research theme *purposed traceability* charts the development of a classification scheme for traceability contexts, and a collection of possible stakeholder requirements on traceability. Also, a "Traceability Book of Knowledge" is planned,

including e.g., terminology, methods, and practices. Furthermore, the research agenda calls for additional empirical studies. Our contribution intensifies CoEST's call for additional industrial case studies, by showing that a majority of IR-based trace recovery studies have been conducted in the "cave of IR evalution". To guide future empirical studies, we propose an adapted version of the model of IR evaluation contexts by Ingwersen and Järvelin [92], tailored for IR-based trace recovery. Also, we confirm the need for a "Traceability Book of Knowledge" and an aligned terminology in the traceability community, as our secondary study was obstructed be language discrepancies.

Trusted traceability comprises research to gain improvements in the quality of creation and maintenance of automatic trace links. Also, the research theme calls for empirical evidence as to the quality of traceability methods and tools with respect to the quality of the trace links. Our work, founded in evidence-based software engineering approaches, aggregated the empirical evidence of IR-based trace recovery until December 2011. Based on this, we provide several advice on how to advance future evaluations.

Finally, the research theme *scalable traceability* calls for the traceability community to obtain and publish large industrial datasets from various domains to enable researchers to investigate scalability of traceability methods. Also this call for research is intensified by our work, as we empirically show that alarmingly few evaluations of IR-based trace recovery have been conducted on industrial datasets of representative sizes.

7 Summary and Future Work

Our review of IR-based trace recovery compares 79 publications containing 132 empirical studies, systematically derived according to established procedures [103]. Our study constitutes the most extensive summary of publications of IR-based trace recovery yet published, enabling an overview of the topic based on empirical results.

More than 10 IR models have been applied to trace recovery (RQ1). More studies have evaluated algebraic IR models (i.e., VSM and LSI), than probabilistic models (e.g., BIM, PIN, LM, LDA). A visible trend is, in line with development in the general field of IR, that the probabilistic subset of statistical language models have received increased attention in recent years. While extracting data from the primary publications, it became clear that the inconsistent use of IR terminology is an issue in the field. In an attempt to homogenize the language, we present structure in the form of a hierarchy of IR models (Figure 6) and a collection of IR terminology (Table 1).

In the 132 mapped empirical studies, artifacts from the entire development process have been linked (RQ2). The dominant artifact type is requirements at various level of abstraction, followed by source code. Substantially fewer studies

have been conducted on test artifacts, and only single publications have targeted user manuals and defect reports. Furthermore, a majority of the evaluations of IR-based trace recovery have been made on bipartite datasets, i.e., only trace links between two disjoint sets of artifacts were recovered.

Among the 79 primary publications mapped in our study, we conclude that the heterogeneity of reporting detail obstructs the aggregation of empirical evidence (RQ3). Also, most evaluations have been conducted on small bipartite datasets containing fewer than 500 artifacts, which is a severe threat to external validity. Furthermore, a majority of evaluations have been using artifacts originating from a university environment, or a dataset of proprietary artifacts from NASA. As a result, the two small datasets EasyClinic and CM-1 constitute the de-facto benchmark in IR-based trace recovery. Another validity threat to the applicability of IR-based trace recovery is that a clear majority of the evaluations have been conducted in "the cave of IR evaluation" as described in Table 2, and reported in Figure 12. Thus, we argue that in-vivo evaluations, in which IR-based trace recovery should be studied within the full complexity of an industrial setting, are needed to motivate the feasibility of the approach and further studies on the topic. As such, our empirical findings intensify the recent call for additional empirical studies by CoEST [79].

Based on our synthesized results from IR-based trace recovery tools, we found no empirical evidence that the technologically-oriented research on tools has resulted in more accurate trace recovery (RQ4). No IR model regularly outperforms the classic VSM with TFIDF feature weighting on text preprocessed by stop word removal and stemming, as presented in Figure 13. As long as trace recovery is based on nothing but the limited NL content of artifacts, there appears to be little value in solely hunting improved P-R values in small datasets, in line with suggestions by Ali *et al.* [4]. Instead, our recommendation is to focus on the evaluations rather than the technical details of the tools. Simply evaluating VSM for trace recovery, the classical IR model with several available open source implementations, in an industrial context has a large potential to contribute to the field.

The strongest empirical evidence in favor of IR-based trace recovery tools comes from a set of controlled experiments on student subjects, reporting that tool-supported subjects outperform manual control groups. However, our results show that only a quarter of the reported P-R values in the primary publications reach 'acceptable' level as defined by Huffman Hayes *et al.* [88]. This suggests that more research is required on how accurate candidate trace links need to be for an engineer to benefit from them, as has been investigated by Cuddeback *et al.* [45], Dekhtyar *et al.* [61], and Borg and Pfahl [22].

In several primary publications it is not made clear whether a query-based or matrix-based evaluation style has been used. Also, the different reporting styles of P-R values make aggregation of candidate trace link accuracies challenging. We argue that the standard measures precision at fixed recall levels and P-R at specific document cut-offs should be reported, complemented by secondary measures such

as MAP and DCG. Moreover, based on P-R values extracted from the query-based evaluations in the primary publications, we show that IR-based trace recovery is considerably more sensitive to the choice of input dataset than to the applied IR model.

As a continuation of this literature study, we intend to publish the extracted data to allow for collaborative editing, and for interested readers to review the details. A possible future study would be to conduct a deeper analysis of the enhancement strategies that have been reported as successful in the primary publications, to investigate patterns concerning in which contexts they have been successfully applied. Also, future work could include categorizing the primary publications according to other frameworks, such as positioning them related to the CoEST research themes.

Acknowledgement

This work was funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering[5]. Thanks go to our librarian Mats Berglund for working on the search strings, and Lorand Dali for excellent comments on IR details.

[5] http://ease.cs.lth.se

Appendix

Tables 8-15 present our classification of the primary publications, sorted by number of citations according to Google Scholar (July 1, 2012). Note that the well-cited works by Marcus and Maletic [120] (354 citations) and Antoniol *et al.* [5] (85 citations) are not listed.

Datasets are classified according to origin: proprietary (Ind), open source (OS), university (Univ), student (Stud), not clearly reported (Unclear), and mixed origin (Mixed). Numbers in parentheses show the number of artifacts studied, a '?' is used when it is not reported. Unless the full dataset name is presented, the following abbreviations are used: IBS (Ice Breaker System), EBT (Event-Based Traceability), LC (Light Control system), TM (Transient Meter). Evaluation, the rightmost column, maps primary publications to the context taxonomy described in Section 3 (Level 1-4 = retrieval context, seeking context, work task context, project context). Finally, Tables 16 show the distinctly most productive authors and affiliations, based upon our primary publications.

Cit.	Title	Authors	IR mod.	Dataset	Evaluation
486	Recovering traceability links between code and documentation	Antoniol, Canfora, De Lucia, Merlo	BIM, VSM	Univ: LEDA (296), Stud: Albergate (116)	Level 1, Level 3 (8 subj.)
205	Advancing candidate link tracing Generation for requirements: The study of methods	Huffman Hayes, Dekhtyar, Sundaram	VSM, LSI	Ind: MODIS (68), CM-1 (455)	Level 2
169	Improving requirements tracing via information retrieval	Huffman Hayes, Dekhtyar, Osborne	VSM	Ind: MODIS (68)	Level 1
140	Recovering traceability links in systems using information retrieval methods	De Lucia, Fasano, Oliveto, Tortora	LSI	Stud: (Multiple projects)	Level 4 (150 subj.)
99	Utilizing supporting evidence to improve dynamic requirements traceability	Cleland-Huang, Settimi, Duan, Zou	PIN	Univ: IBS (252), EBT (114), LC (61)	Level 1
79	Best practices for automated traceability	Cleland-Huang, Berenbach, Clark, Settimi, Romanova	PIN	Ind: Siemens Logistics and Automation (?), Univ: IBT (255), EBT (114)	Level 1
74	Helping analysts trace requirements: An objective look	Huffman Hayes, Dekhtyar, Sundaram, Howard	VSM	Ind: MODIS (68)	Level 2
70	Can LSI help reconstructing requirements traceability in design and test?	Lormans, van Deursen	LSI	Ind: Philips (359), Stud: PacMan (46), Callisto (?)	Level 1
68	Supporting software evolution through dynamically retrieving traces to UML artifacts	Settimi, Cleland-Huang, Khadra, Mody, Lukasik, DePalma	VSM	Univ: EBT (138)	Level 1
64	Enhancing an artefact management system with traceability recovery Features	De Lucia, Fasano, Oliveto, Tortora	LSI	Stud: EasyClinic (150)	Level 1

Table 8: Classification of primary publications, part I.

Cit.	Title	Authors	IR mod.	Dataset	Evaluation
58	Recovery of traceability links between software documentation and source code	Marcus, Maletic, Sergeyev	LSI	Univ: LEDA (228-803), Stud: Albergate (73)	Level 1
44	Recovering code to documentation links in OO systems	Antoniol, Canfora, De Lucia, Marlo	BIM	Univ: LEDA (296)	Level 1
40	Fine grained indexing of software repositories to support impact analysis	Canfora, Cerulo	BM25	OS: Gedit (233), ArgoUML (2208), Firefox (680)	Level 1
38	ADAMS Re-Trace: A traceability recovery tool	De Lucia, Fasano, Oliveto, Tortora	LSI	Stud: (48, 50, 54, 55, 73, 74, 111)	Level 4 (7 proj.)
36	On the equivalence of information retrieval methods for automated traceability link recovery	Oliveto, Gethers, Poshyvanyk, De Lucia	VSM, LSI, LM, LDA	Stud: EasyClinic (77), eTour (174)	Level 1
33	Incremental approach and user feedbacks: A silver bullet for traceability recovery	De Lucia, Oliveto, Sgueglia	VSM, LSI	Ind: MODIS (68), Stud: EasyClinic (150)	Level 1
30	A machine learning approach for tracing regulatory codes to product specific requirements	Cleland-Huang, Czauderna, Gibiec, Emenecker	PIN	Mixed: (254)	Level 2
30	Assessing IR-based traceability recovery tools through controlled experiments	De Lucia, Oliveto, Tortora	LSI	Stud: EasyClinic (150)	Level 3 (20, 12 subj.)
29	A traceability technique for specifications	Abadi, Nisenson, Simionovici	VSM, LSI, PLSI, SDR, LM	OS: SCA (1311), CORBA (3340)	Level 2
29	Can information retrieval techniques effectively support traceability link recovery?	De Lucia, Fasano, Oliveto, Tortora	LSI	Stud: EasyClinic (150), Univ: ADAMS (309), LEDA (803)	Level 1, Level 4 (150 subj.)

Table 9: Classification of primary publications, part II.

Cit.	Title	Authors	IR mod.	Dataset	Evaluation
29	Software traceability with topic modeling	Asuncion, Asuncion, Taylor	LSI, LDA	Univ: ArchStudio (?), Stud: EasyClinic (160)	Level 1
29	Speeding up requirements to management in a product software company: Linking customer wishes to product requirements through linguistic engineering	Natt och Dag, Gervasi, Brinkkemper, Regnell	VSM	Ind: Baan (12083)	Level 2
29	Tracing object-oriented code into functional requirements	Antoniol, Canfora, De Lucia, Casazza, Merlo	BIM	Stud: Albergate (76)	Level 1
28	Clustering support for automated tracing	Duan, Cleland-Huang	PIN	Univ: IBS (185)	Level 1
27	Text mining for software engineering: how analyst feedback impacts final results	Huffman Hayes, Dekhtyar, Sundaram	N/A	Ind: MODIS (68)	Level 3 (3 subj.)
26	A feasibility study of automated natural language requirements analysis in market-driven development	Natt och Dag, Regnell, Carlshamre, Andersson, Karlsson	VSM	Ind: Telelogic (1891, 1089)	Level 1
26	Implementation of an efficient requirements analysis supporting system using similarity measure techniques	Park, Kim, Ko, Seo	Sliding window, syntactic parser	Ind: Unclear (33)	Level 1
25	Traceability recovery in RAD software systems	Di Penta, Gradara, Antoniol	BIM	Univ: TM (49)	Level 1
23	REquirements TRacing On target (RETRO): Improving software maintenance through traceability recovery	Huffman Hayes, Dekhtyar, Holbrook, Sundaram, Vadlamudi, April	VSM	Ind: CM-1 (74)	Level 3 (30 subj.)
22	Phrasing in dynamic requirements trace retrieval	Zou, Settimi, Cleland-Huang	PIN	Univ: IBS (235), LC (59), EBT (93)	Level 1

Table 10: Classification of primary publications, part III.

Cit.	Title	Authors	IR mod.	Dataset	Evaluation
21	Combining textual and structural analysis of software artifacts for traceability link recovery	McMillan, Poshyvanyk, Revelle	LSI	Univ: CoffeeMaker (143)	Level 1
20	Tracing requirements to defect reports: An application of information retrieval techniques	Yadla, Huffman Hayes, Dekhtyar	VSM	Ind: CM-1 (68,118)	Level 2
18	Automated requirements traceability: The study of human analysts	Cuddeback, Dekhtyar, Huffman Hayes	VSM	OS: BlueJ Plugin (49)	Level 3 (26 subj.)
18	Incremental latent semantic indexing for automatic traceability link evolution management	Jiang, Nguyen, Chen, Jaygarl, Chang	LSI	Univ: LEDA (634)	Level 1
18	Understanding how the requirements are implemented in source code	Zhao, Zhang, Liu, Juo, Sun	VSM	OS: Desktop Calculator (123)	Level 1
17	Improving automated requirements trace retrieval: A study of term-based enhancement methods	Zou, Settimi, Cleland-Huang	PIN	Ind: CM-1 (455), Univ: IBS (235), EBT (93), LC (89), Stud: SE450 (521)	Level 2
17	IR-based traceability recovery processes: An empirical comparison of "one-shot" and incremental processes	De Lucia, Oliveto, Tortora	LSI	Stud: EasyClinic (150)	Level 3 (30 subj.)
17	Make the most of your time: how should the analyst work with automated traceability tools?	Dekhtyar, Huffman Hayes, Larsen	VSM	Ind: CM-1 (455)	Level 2
16	Baselines in requirements tracing	Sundaram, Huffman Hayes, Dekhtyar	VSM, LSI	Ind: CM-1 (455), MODIS (68)	Level 2
11	Challenges for semi-automatic trace recovery in the automotive domain	Leuser	VSM, LSI	Ind: Daimler AG (1500)	Level 1

Table 11: Classification of primary publications, part IV.

Cit.	Title	Authors	IR mod.	Dataset	Evaluation
11	Monitoring requirements coverage using reconstructed views: an industrial case study	Lormans, Gross, van Deursen, Stehouwer, van Solingen	LSI	Ind: LogicaCMG (219)	Level 1
11	On the role of the nouns in IR-based traceability recovery	Capobianco, De Lucia, Oliveto, Panichella, Panichella	LSI, LM	Stud: EasyClinic (150)	Level 1
10	An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven product development	Natt och Dag, Thelin, Regnell	VSM	Stud: PUSS (299)	Level 3 (23 subj.)
9	An industrial case study in reconstructing requirements views	Lormans, van Deursen, Gross	LSI	Ind: LogicaCMG (293)	Level 1
9	Towards mining replacement queries for hard-to-retrieve traces	Gibiec, Czauderna, Cleland-Huang	VSM	Mixed: (254)	Level 2
8	Recovering relationships between documentation and source code based on the characteristics of software engineering	Wang, Lai, Liu	LSI, BIM	Univ: LEDA (597), IBS (270)	Level 1
8	Trace retrieval for evolving artifacts	Winkler	LSI	Ind: Robert Bosch GmbH (500), MODIS (68)	Level 1
8	Traceability recovery using numerical analysis	Capobianco, De Lucia, Oliveto, Panichella, Panichella	VSM, LSI, LM, B-splines	Stud: EasyClinic (150)	Level 1
7	Assessing traceability of software engineering artifacts	Sundaram, Huffman Hayes, Dekhtyar, Holbrook	VSM, LSI	Ind: MODIS (68), CM-1 (455), Stud: 22* Waterloo (65)	Level 2
7	Requirement-centric traceability for change impact analysis: A case study	Li, Li, Yang, Li	VSM	Unclear: Requirements Management System (501)	Level 4 (5 subj.)

Table 12: Classification of primary publications, part V.

Cit.	Title	Authors	IR mod.	Dataset	Evaluation
6	How do we trace requirements: An initial study of analyst behavior in trace validation tasks	Kong, Huffman Hayes, Dekhtyar, Holden	N/A	OS: BlueJ plugin (49)	Level 3 (13 subj.)
6	Technique integration for requirements assesment	Dekhtyar, Huffman Hayes, Sundaram, Holbrook, Dekhtyar	VSM, LSI, BIM, LDA, Chi2 key extr.	Ind: CM-1 (455)	Level 1
4	Application of swarm techniques for requirements engineering: Requirements tracing	Sultanov, Huffman Hayes	VSM, Swarm	Ind: CM-1 (455), Univ: PINE (182)	Level 1
4	On integrating orthogonal information retrieval methods to improve traceability recovery	Gethers, Oliveto, Posyvanyk, De Lucia	VSM, LM, RTM	Stud: eAnsi (194), eAnsi (67), EasyClinic (57), EasyClinic (100), eTour (232), SMOS (167)	Level 1
3	A clustering-based approach for tracing object-oriented design to requirement	Zhou, Yu	VSM	Univ: Resource Management Software (33)	Level 1
3	Evaluating the use of project glossaries in automated trace retrieval	Zou, Settimi, Cleland-Huang	PIN	Ind: CM-1 (455), Univ: IBS (235), Stud: SE450 (61)	Level 1
3	On human analyst performance in assisted requirements tracing: Statistical analysis	Dekhtyar, Dekhtyar, Holden, Huffman Hayes, Cuddeback, Kong	VSM	OS: BlueJ (49)	Level 3 (84 subj.)
3	Tackling semi-automatic trace recovery for large specifications	Leuser, Ott	VSM	Ind: Daimler (2095, 944)	Level 1
2	Extraction and visualization of traceability relationships between documents and source code	Chen	Unclear	OS: JDK1.5 (?), uDig 1.1.1 (?)	Level 1
2	Source code indexing for automated tracing	Mahmoud, Niu	VSM	Stud: eTour (174), iTrust (264)	Level 1

Table 13: Classification of primary publications, part VI.

Cit.	Title	Authors	IR mod.	Dataset	Evaluation
2	Traceability challenge 2011: Using Tracelab to evaluate the impact of local versus global IDF on trace retrieval	Czauderna, Gibiec, Leach, Li, Shin, Keenan, Cleland-Huang	VSM	Ind: CM-1 (75), WV-CCHIT (1180)	Level 2
2	Trust-based requirements traceability	Ali, Gueheneuc, Antoniol	VSM	OS: Pooka (388), SIP (1853)	Level 1
1	An adaptive approach to impact analysis from change requests to source code	Gethers, Kagdi, Dit, Poshyvanyk	LSI	OS: ArgoUML (qualitative analysis)	Level 2
1	Do better IR tools improve the accuracy of engineers' traceability recovery?	Borg, Pfahl	VSM	Ind: CM-1 (455)	Level 3 (8 subj.)
1	Experiences with text mining large collections of unstructured systems development artifacts at JPL	Port, Nikora, Hihn, Huang	LSI	Unclear	Level 3
1	Improving automated documentation to code traceability by combining retrieval techniques	Chen, Grundy	VSM	OS: JDK (431)	Level 1
1	Improving IR-based traceability recovery using smoothing filters	De Lucia, Di Penta, Oliveto, Panichella, Panichella	VSM, LSI	Univ: PINE (131), Stud: EasyClinic (150)	Level 1
1	Using semantics-enabled information retrieval in requirements tracing: An ongoing experimental investigation	Mahmoud, Niu	VSM	Ind: CM-1 (455)	Level 1
1	Traceclipse: An eclipse plug-in for traceability link recovery and management	Klock, Gethers, Dit, Poshyvanyk	Unclear	Ind: CM-1 (455), Stud: EasyClinic (150)	Level 1
0	A combination approach for enhancing automated traceability (NIER track)	Chen, Hosking, Grundy	VSM	OS: JDK 1.5 (?)	Level 1

Table 14: Classification of primary publications, part VII.

Cit.	Title	Authors	IR mod.	Dataset	Evaluation
0	A comparative study of document correlation techniques for traceability analysis	Parvathy, Vasudevan, Balakrishnan	VSM, LSI, LDA, CTM	Unclear: (43), (261)	Level 1
0	A requirement traceability refinement method based on relevance feedback	Kong, Li, Li, Yang, Wang	VSM, LM	Ind: Web app (511)	Level 1
0	An improving approach for recovering requirements-to-design traceability links	Di, Zhang	BIM	Ind: CM-1 (455), MODIS (68)	Level 1
0	Proximity-based traceability: An empirical validation using ranked retrieval and set-based measures	Kong, Huffman Hayes	VSM	Ind: CM-1 (75), OS: Pine (182), Univ: StyleChecker (49), Stud: EasyClinic (77)	Level 2
0	Reconstructing traceability between bugs and test cases: An experimental study	Kaushik, Tahvildari, Moore	LSI	Ind: RIM (13389)	Level 1
0	Requirements traceability for object oriented systems by partitioning source code	Ali, Guehenuec, Antoniol	VSM	OS: Pooka (388), SIP (1853), Univ: iTrust (526)	Level 1
0	Software verification and validation research laboratory (SVVRL) of the University of Kentucky: Traceability challenge 2011: Language translation	Huffman Hayes, Sultanov, Kong, Li	VSM	Stud: EasyClinic (150), eTour (174)	Level 2
0	The role of the coverage analysis during IR-based traceability recovery: A controlled experiment	De Lucia, Oliveto, Tortora	LSI	Stud: EasyClinic (150)	Level 3 (30 subj.)
0	Towards a benchmark for traceability	Ben Charrada, Casper, Jeanneret, Glinz	VSM	Univ: AquaLush (793)	Level 1

Table 15: Classification of primary publications, part VIII.

Author	**Publications**
Andrea De Lucia	16 (9)
Jane Huffman Hayes	16 (6)
Alexander Dekhtyar	15 (3)
Rocco Oliveto	13 (1)
Jane Cleland-Huang	10 (3)
Affiliation	**Publications**
University of Kentucky, United States	13
University of Salerno, Italy	11
DePaul University, United States	10
University of Sannio, Italy	5

Table 16: Most productive authors and affiliations. For authors, the first number is the total number of primary publications, while the number in parenthesis is first-authored primary publications. For affiliations, the numbers show the number of primary publications first-authored by an affiliated researcher.

Bibliography

[1] A. Abadi, M. Nisenson, and Y. Simionovici. A traceability technique for specifications. In *Proceedings of the 16th International Conference on Program Comprehension*, pages 103–112, 2008.

[2] N. Ali, Y-G. Guéhéneuc, and G. Antoniol. Requirements traceability for object oriented systems by partitioning source code. In *Proceedings of the 18th Working Conference on Reverse Engineering*, pages 45–54, 2011.

[3] N. Ali, Y-G. Guéhéneuc, and G. Antoniol. Trust-based requirements traceability. In *Proceedings of the 19th International Conference on Program Comprehension*, pages 111–120, 2011.

[4] N. Ali, Y-G. Guéhéneuc, and G. Antoniol. Factors impacting the inputs of traceability recovery approaches. In J. Cleland-Huang, O. Gotel, and A. Zisman, editors, *Software and Systems Traceability*. Springer, 2012.

[5] G. Antoniol, G. Canfora, G. Casazza, and A. De Lucia. Information retrieval models for recovering traceability links between code and documentation. In *Conference on Software Maintenance*, pages 40–49, 2000.

[6] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Tracing object-oriented code into functional requirements. In *Proceedings of the 8th International Workshop on Program Comprehension*, pages 79–86, 2000.

[7] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering traceability links between code and documentation. In *Transactions on Software Engineering*, volume 28, pages 970–983, 2002.

[8] G. Antoniol, G. Canfora, A. De Lucia, and E. Merlo. Recovering code to documentation links in OO systems. In *Proceedings of the 6th Working Conference on Reverse Engineering*, pages 136–144, 1999.

[9] G. Antoniol, A. Potrich, P. Tonella, and R. Fiutem. Evolving object oriented design to improve code traceability. In *Proceedings of the 7th International Workshop on Program Comprehension*, pages 151–160, 1999.

[10] N. Assawamekin, T. Sunetnanta, and C. Pluempitiwiriyawej. Ontology-based multiperspective requirements traceability framework. *Knowledge and Information Systems*, 25(3):493–522, 2010.

[11] H. Asuncion, A. Asuncion, and R. Taylor. Software traceability with topic modeling. In *Proceedings of the International Conference on Software Engineering*, pages 95–104, 2010.

[12] K. Ayari, P. Meshkinfam, G. Antoniol, and M. Di Penta. Threats on building models from CVS and Bugzilla repositories: The Mozilla case study. In *Proceedings of the Conference of the Center for Advanced Studies on Collaborative Research*, pages 215–228, 2007.

[13] A. Bacchelli, M. Lanza, and R. Robbes. Linking e-mails and source code artifacts. In *Proceedings of the 32nd International Conference on Software Engineering*, pages 375–384, 2010.

[14] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval: The concepts and technology behind search*. Addison-Wesley, 2nd edition, 2011.

[15] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 26–33, 2001.

[16] E. Ben Charrada, D. Caspar, C. Jeanneret, and M. Glinz. Towards a benchmark for traceability. In *Proceedings of the 12th International Workshop on Principles on Software Evolution*, pages 21–30, 2011.

[17] A. Bianchi, A. Fasolino, and G. Visaggio. An exploratory case study of the maintenance effectiveness of traceability models. In *Proceedings of the 8th International Workshop on Program Comprehension*, pages 149–158, 2000.

[18] D. Binkley and D. Lawrie. Information retrieval applications in software maintenance and evolution. In J. Marciniak, editor, *Encyclopedia of software engineering*. Taylor & Francis, 2nd edition, 2010.

[19] D. Blei and J. Lafferty. A correlated topic model of science. *Annals of Applied Statistics*, 1(1):17–35, 2007.

[20] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3(4-5):993–1022, 2003.

[21] M. Borg. Findability through traceability: A realistic application of candidate trace links? In *Proceedings of the 7th International Conference on Evaluating Novel Approaches to Software Engineering*, pages 173–181, 2012.

[22] M. Borg and D. Pfahl. Do better IR tools improve the accuracy of engineers' traceability recovery? In *Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering*, pages 27–34, 2011.

[23] M. Borg, P. Runeson, and L. Brodén. Evaluation of traceability recovery in context: A taxonomy for information retrieval tools. In *Proceedings of the 16th International Conference on Evaluation & Assessment in Software Engineering*, pages 111–120, 2012.

[24] M. Borg, K. Wnuk, and D. Pfahl. Industrial comparability of student artifacts in traceability recovery research - an exploratory survey. In *Proceedings of the 16th European Conference on Software Maintenance and Reengineering*, pages 181–190, 2012.

[25] M. Borillo, A. Borillo, N. Castell, D. Latour, Y. Toussaint, and M. Felisa Verdejo. Applying linguistic engineering to spatial software engineering: The traceability problem. In *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 593–595, 1992.

[26] M. Bras and Y. Toussaint. Artificial intelligence tools for software engineering: Processing natural language requirements. In *Applications of Artificial Intelligence in Engineering*, pages 275–290, 1993.

[27] P. Brereton, B. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571–583, 2007.

[28] G. Canfora and L. Cerulo. Fine grained indexing of software repositories to support impact analysis. In *Proceedings of the International Workshop on Mining software repositories*, pages 105–111, 2006.

[29] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, and S. Panichella. On the role of the nouns in IR-based traceability recovery. In *Proceedings of the 17th International Conference on Program Comprehension*, pages 148–157, 2009.

[30] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, and S. Panichella. Traceability recovery using numerical analysis. In *Proceedings of the 16th Working Conference on Reverse Engineering*, pages 195–204, 2009.

[31] Carnegie Mellon Software Engineering Institute. CMMI for development, Version 1.3, 2010.

[32] N. Castell, O. Slavkova, Y. Toussaint, and A. Tuells. Quality control of software specifications written in natural language. In *Proceedings of the 7th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 37–44, 1994.

[33] J. Chang and D. Blei. Hierarchical relational models for document networks. *The Annals of Applied Statistics*, 4(1):124–150, 2010.

[34] X. Chen. Extraction and visualization of traceability relationships between documents and source code. In *Proceedings of the International Conference on Automated Software Engineering*, pages 505–509, 2010.

[35] X. Chen, J. Hosking, and J. Grundy. A combination approach for enhancing automated traceability. In *Proceeding of the 33rd International Conference on Software Engineering, (NIER track)*, pages 912–915, 2011.

[36] J. Cleland-Huang, C. K Chang, and M. Christensen. Event-based traceability for managing evolutionary change. *Transactions on Software Engineering*, 29(9):796– 810, 2003.

[37] J. Cleland-Huang, A. Czauderna, A. Dekhtyar, O. Gotel, J. Huffman Hayes, E. Keenan, J. Maletic, D. Poshyvanyk, Y. Shin, A. Zisman, G. Antoniol, B. Berenbach, A. Egyed, and P. Mäder. Grand challenges, benchmarks, and TraceLab: Developing infrastructure for the software traceability research community. In *Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*, 2011.

[38] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker. A machine learning approach for tracing regulatory codes to product specific requirements. In *Proceedings International Conference on Software Engineering*, pages 155–164, 2010.

[39] J. Cleland-Huang, O. Gotel, and A. Zisman, editors. *Software and systems traceability*. Springer, 2012.

[40] J. Cleland-Huang, J. Huffman Hayes, and A. Dekhtyar. Center of excellence for traceability: Problem statement and grand challenges in traceability (v0.1). Technical Report COET-GCT-06-01-0.9, 2006.

[41] J. Cleland-Huang, W. Marrero, and B. Berenbach. Goal-centric traceability: Using virtual plumblines to maintain critical systemic qualities. *Transactions on Software Engineering*, 34(5):685–699, 2008.

[42] J. Cleland-Huang, R. Settimi, C. Duan, and X. C. Zou. Utilizing supporting evidence to improve dynamic requirements traceability. In *Proceedings of the 13th International Conference on Requirements Engineering*, pages 135–144, 2005.

[43] J. Cleland-Huang, R. Settimi, E. Romanova, B. Berenbach, and S. Clark. Best practices for automated traceability. *Computer*, 40(6):27–35, 2007.

[44] C. Cleverdon. The significance of the Cranfield tests on index languages. In *Proceedings of the 14th Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1991.

[45] D. Cuddeback, A. Dekhtyar, and J. Huffman Hayes. Automated requirements traceability: The study of human analysts. In *Proceedings of the 18th International Requirements Engineering Conference*, pages 231–240, 2010.

[46] A. Czauderna, M. Gibiec, G. Leach, Y. Li, Y. Shin, E. Keenan, and J. Cleland-Huang. Traceability challenge 2011: Using Tracelab to evaluate the impact of local versus global IDF on trace retrieval. In *Proceeding of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*, pages 75–78, 2011.

[47] A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella. Improving IR-based traceability recovery using smoothing filters. In *Proceedings of the 19th International Conference on Program Comprehension*, pages 21–30, 2011.

[48] A. De Lucia, M. Di Penta, R. Oliveto, and F. Zurolo. COCONUT: COde COmprehension nurturant using traceability. In *Proceedings of the 22nd International Conference on Software Maintenance*, pages 274–275, 2006.

[49] A. De Lucia, M. Di Penta, R. Oliveto, and F. Zurolo. Improving comprehensibility of source code via traceability information: A controlled experiment. In *Proceedings of the International Conference on Program Comprehension*, pages 317–326, 2006.

[50] A. De Lucia, F. Fasano, and R. Oliveto. Traceability management for impact analysis. In *Frontiers of Software Maintenance*, pages 21–30, 2008.

[51] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Enhancing an artefact management system with traceability recovery features. In *Proceedings of the 20th International Conference on Software Maintenance*, pages 306–315, 2004.

[52] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. ADAMS re-trace: A traceability recovery tool. In *Proceedings of the 9th European Conference on Software Maintenance and Reengineering*, pages 32–41, 2005.

[53] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Can information retrieval techniques effectively support traceability link recovery? In *Proceedings of the 14th International Conference on Program Comprehension*, pages 307–316, 2006.

[54] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Recovering traceability links in software artifact management systems using information retrieval methods. *Transactions on Software Engineering and Methodology*, 16(4), 2007.

[55] A. De Lucia, A. Marcus, R. Oliveto, and D. Poshyvanyk. Information retrieval methods for automated traceability recovery. In J. Cleland-Huang, O. Gotel, and A. Zisman, editors, *Software and Systems Traceability*. Springer, 2012.

[56] A. De Lucia, R. Oliveto, and P. Sgueglia. Incremental approach and user feedbacks: A silver bullet for traceability recovery? In *Proceedings of the International Conference on Software Maintenance*, pages 299–308, 2006.

[57] A. De Lucia, R. Oliveto, and G. Tortora. IR-based traceability recovery processes: An empirical comparison of "one-shot" and incremental processes. In *Proceedings of the 23rd International Conference on Automated Software Engineering*, pages 39–48, 2008.

[58] A. De Lucia, R. Oliveto, and G. Tortora. Assessing IR-based traceability recovery tools through controlled experiments. *Empirical Software Engineering*, 14(1):57–92, 2009.

[59] A. De Lucia, R. Oliveto, and G. Tortora. The role of the coverage analysis during IR-based traceability recovery: A controlled experiment. In *Proceedings of the International Conference on Software Maintenance*, pages 371–380, 2009.

[60] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[61] A. Dekhtyar, O. Dekhtyar, J. Holden, J. Huffman Hayes, D. Cuddeback, and W. Kong. On human analyst performance in assisted requirements tracing: Statistical analysis. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 111–120, 2011.

[62] A. Dekhtyar and J. Huffman Hayes. Good benchmarks are hard to find: Toward the benchmark for information retrieval applications in software engineering. *Proceedings of the International Conference on Software Maintenance*, 2006.

[63] A. Dekhtyar, J. Huffman Hayes, and G. Antoniol. Benchmarks for traceability? In *Proceedings of the International Symposium on Grand Challenges in Traceability*, 2007.

[64] A. Dekhtyar, J. Huffman Hayes, and J. Larsen. Make the most of your time: How should the analyst work with automated traceability tools? In *Proceedings of the 3rd International Workshop on Predictor Models in Software Engineering*, 2007.

[65] A. Dekhtyar, J. Huffman Hayes, S. Sundaram, A. Holbrook, and O. Dekhtyar. Technique integration for requirements assessment. In *Proceedings of the 15th International Requirements Engineering Conference*, pages 141–152, 2007.

[66] F. Di and M. Zhang. An improving approach for recovering requirements-to-design traceability links. In *Proceedings of the International Conference on Computational Intelligence and Software Engineering*, pages 1–6, 2009.

[67] M. Di Penta, S. Gradara, and G. Antoniol. Traceability recovery in RAD software systems. In *Proceedings of the 10th International Workshop on Program Comprehension*, pages 207–216, 2002.

[68] B. Dit, M. Revelle, M. Gethers, and D. Poshyvanyk. Feature location in source code: A taxonomy and survey. *Journal of Software Maintenance and Evolution: Research and Practice*, 2011.

[69] R. Dömges and K. Pohl. Adapting traceability environments to project-specific needs. *Communications of the ACM*, 41(12):54–62, 1998.

[70] C. Duan and J. Cleland-Huang. Clustering support for automated tracing. In *Proceedings of the International Conference on Automated Software Engineering*, pages 244–253, 2007.

[71] A. Egyed and P. Grünbacher. Automating requirements traceability: Beyond the record replay paradigm. In *Proceedings of the 17th International Conference on Automated Software Engineering*, pages 163–171, 2002.

[72] T. Eisenbarth, R. Koschke, and D. Simon. Locating features in source code. *Transactions on Software Engineering*, 29(3):210– 224, 2003.

[73] D. Falessi, G. Cantone, and G. Canfora. A comprehensive characterization of NLP techniques for identifying equivalent requirements. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, 2010.

[74] K. R Felizardo, N. Salleh, R. M Martins, E. Mendes, S. G MacDonell, and J. C Maldonado. Using visual text mining to support the study selection activity in systematic literature reviews. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, pages 77–86, 2011.

[75] R. Fiutem and G. Antoniol. Identifying design-code inconsistencies in object-oriented software: A case study. In *Proceedings of the International Conference on Software Maintenance*, pages 94–102, 1998.

[76] G. Gay, S. Haiduc, A. Marcus, and T. Menzies. On the use of relevance feedback in IR-based concept location. In *Proceedings of the 25th International Conference on Software Maintenance*, pages 351–360, 2009.

[77] M. Gethers, R. Oliveto, D. Poshyvanyk, and A. De Lucia. On integrating orthogonal information retrieval methods to improve traceability recovery. In *Proceedings of the International Conference on Software Maintenance*, pages 133–142, 2011.

[78] M. Gibiec, A. Czauderna, and J. Cleland-Huang. Towards mining replacement queries for hard-to-retrieve traces. In *Proceedings of the International Conference on Automated Software Engineering*, pages 245–254, 2010.

[79] O. Gotel, J. Cleland-Huang, J. Huffman Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, and J. Maletic. The grand challenge of traceability (v1.0). In J. Cleland-Huang, O. Gotel, and A. Zisman, editors, *Software and Systems Traceability*. Springer, 2012.

[80] O. Gotel and C. Finkelstein. An analysis of the requirements traceability problem. In *Proceedings of the First International Conference on Requirements Engineering*, pages 94–101, 1994.

[81] M. Heindl and S. Biffl. A case study on value-based requirements tracing. In *Proceedings of the 10th European Software Engineering Conference held jointly with the 13th SIGSOFT International Symposium on Foundations of Software Engineering*, pages 60–69, 2005.

[82] T. Hofman. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196, 2001.

[83] J. Huffman Hayes, G. Antoniol, and Y-G. Guéhéneuc. PREREQIR: recovering pre-requirements via cluster analysis. In *Proceedings of the 15th Working Conference on Reverse Engineering*, pages 165–174, 2008.

[84] J. Huffman Hayes and A. Dekhtyar. A framework for comparing requirements tracing experiments. *Interational Journal of Software Engineering and Knowledge Engineering*, 15(5):751–781, 2005.

[85] J. Huffman Hayes and A. Dekhtyar. Humans in the traceability loop: Can't live with 'em, can't live without 'em. In *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering*, pages 20–23, 2005.

[86] J. Huffman Hayes, A. Dekhtyar, and J. Osborne. Improving requirements tracing via information retrieval. In *Proceedings of the 11th Internationl Requirements Engineering Conference*, pages 138–147, 2003.

[87] J. Huffman Hayes, A. Dekhtyar, and S. Sundaram. Text mining for software engineering: How analyst feedback impacts final results. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 1–5, 2005.

[88] J. Huffman Hayes, A. Dekhtyar, and S. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *Transactions on Software Engineering*, 32(1):4–19, 2006.

[89] J. Huffman Hayes, A. Dekhtyar, S. Sundaram, A. Holbrook, S. Vadlamudi, and A. April. REquirements TRacing on target (RETRO): improving software maintenance through traceability recovery. *Innovations in Systems and Software Engineering*, 3(3):193–202, 2007.

[90] J. Huffman Hayes, A. Dekhtyar, S. Sundaram, and S. Howard. Helping analysts trace requirements: An objective look. In *Proceedings of the International Conference on Requirements Engineering*, pages 249–259, 2004.

[91] J. Huffman Hayes, H. Sultanov, W. Kong, and W. Li. Software verification and validation research laboratory (SVVRL) of the University of Kentucky: Traceability challenge 2011: Language translation. In *Proceeding of the 6th international workshop on Traceability in emerging forms of software engineering*, pages 50–53. ACM, 2011.

[92] P. Ingwersen and K. Järvelin. *The turn: Integration of information seeking and retrieval in context*. Springer, 2005.

[93] International Electrotechnical Commission. IEC 61511-1 ed 1.0, Safety instrumented systems for the process industry sector, 2003.

[94] International Organization for Standardization. ISO 26262-1:2011 Road vehicles –Functional safety –, 2011.

[95] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48, 2000.

[96] A. Jedlitschka, M. Ciolkowski, and D. Pfahl. Reporting experiments in software engineering. In F. Shull, J. Singer, and D. Sjöberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 201–228. Springer, London, 2008.

[97] H. Jiang, T. Nguyen, I. Chen, H. Jaygarl, and C. Chang. Incremental latent semantic indexing for automatic traceability link evolution management. In *Proceedings of the 23rd International Conference on Automated Software Engineering*, pages 59–68, 2008.

[98] N. Kando. NTCIR workshop : Japanese- and Chinese-English cross-lingual information retrieval and multi-grade relevance judgments. In *Cross-Language Information Retrieval and Evaluation*, volume 2069, pages 24–35. 2001.

[99] V. Katta and T. Stålhane. A conceptual model of traceability for safety systems. In *Proceedings of the Complex Systems Design & Management Conference*, 2011.

[100] N. Kaushik, L. Tahvildari, and M. Moore. Reconstructing traceability between bugs and test cases: An experimental study. In *Proceedings of the Working Conference on Reverse Engineering*, pages 411–414, 2011.

[101] J. Kekäläinen and K. Järvelin. Evaluating information retrieval systems under the challenges of interaction and multidimensional dynamic relevance. *Proceedings of the COLIS 4 Conference*, pages 253—270, 2002.

[102] B. Kitchenham, D. Budgen, and P. Brereton. Using mapping studies as the basis for further research - A participant-observer case study. *Information and Software Technology*, 53(6):638–651, 2011.

[103] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. *EBSE Technical Report*, 2007.

[104] S. Klock, M. Gethers, B. Dit, and D. Poshyvanyk. Traceclipse: An eclipse plug-in for traceability link recovery and management. In *Proceeding of the 6th International Workshop on Traceability in Emerging Forms of Software Eengineering*, pages 24–30, 2011.

[105] L. Kong, J. Li, Y. Li, Y. Yang, and Q. Wang. A requirement traceability refinement method based on relevance feedback. In *Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering*, 2009.

[106] P. Kruchten. *The Rational Unified Process: An introduction*. Addison-Wesley Professional, 2004.

[107] J. Leuser. Challenges for semi-automatic trace recovery in the automotive domain. In *Proceedings of the International Workshop on Traceability in Emerging Forms of Software Engineering*, pages 31–35, 2009.

[108] J. Leuser and D. Ott. Tackling semi-automatic trace recovery for large specifications. In *Requirements Engineering: Foundation for Software Quality*, pages 203–217, 2010.

[109] D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Machine Learning: ECML-98*, volume 1398, pages 4–15. Springer, 1998.

[110] Y. Li, J. Li, Y. Yang, and M. Li. Requirement-centric traceability for change impact analysis: A case study. In *International Conference on Software Process*, pages 100–111, 2008.

[111] E. Liddy. *Natural language processing*. Encyclopedia of Library and Information Science. Marcel Decker, 2nd edition, 2001.

[112] J. Lin, L. Chan, J. Cleland-Huang, R. Settimi, J. Amaya, G. Bedford, B. Berenbach, O. B Khadra, D. Chuan, and X. Zou. Poirot: A distributed tool supporting enterprise-wide automated traceability. In *Proceedings of the 14th International Conference on Requirements Engineering*, pages 363–364, 2006.

[113] M. Lindvall, R. Feldmann, G. Karabatis, Z. Chen, and V. Janeja. Searching for relevant software change artifacts using semantic networks. In *Proceedings of the Symposium on Applied Computing*, pages 496–500, 2009.

[114] M. Lormans, H-G. Gross, A. van Deursen, R. van Solingen, and A. Stehouwer. Monitoring requirements coverage using reconstructed views: An industrial case study. In *Procedings of the 13th Working Conference on Reverse Engineering*, pages 275–284, 2006.

[115] M. Lormans and A. van Deursen. Can LSI help reconstructing requirements traceability in design and test? In *Proceedings of the 10th European Conference on Software Maintenance and Reengineering*, pages 45–54, 2006.

[116] M. Lormans, A. Van Deursen, and H-G. Gross. An industrial case study in reconstructing requirements views. *Empirical Software Engineering*, 13(6):727–760, 2008.

[117] A. Mahmoud and N. Niu. Using semantics-enabled information retrieval in requirements tracing: An ongoing experimental investigation. In *Proceedings of the International Computer Software and Applications Conference*, pages 246–247, 2010.

[118] A. Mahmoud and N. Niu. Source code indexing for automated tracing. In *Proceeding of the 6th International Workshop on Traceability in Emerging forms of Software Engineering*, pages 3–9, 2011.

[119] C. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[120] A. Marcus and J. Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. In *Proceedings of the International Conference on Software Engineering*, pages 125–135, 2003.

[121] A. Marcus, J. Maletic, and A. Sergeyev. Recovery of traceability links between software documentation and source code. *International Journal of Software Engineering and Knowledge Engineering*, 15(5):811–836, 2005.

[122] A. Marcus, A. Sergeyev, V. Rajlich, and J. I. Maletic. An information retrieval approach to concept location in source code. In *Proceedings of the 11th Working Conference on Reverse Engineering*, pages 214–223, 2004.

[123] A. Marcus, X. Xie, and D. Poshyvanyk. When and how to visualize traceability links? In *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering*, pages 56–61, 2005.

[124] M. Maron and J. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM*, 7(3):216–244, 1960.

[125] C. McMillan, D. Poshyvanyk, and M. Revelle. Combining textual and structural analysis of software artifacts for traceability link recovery. In *Proceedings of the International Workshop on Traceability in Emerging Forms of Software Engineering*, pages 41–48, 2009.

[126] M. Miles and M. Huberman. *Qualitative data analysis: An expanded sourcebook*. Sage Publications, 2nd edition, 1994.

[127] P. Morville. *Ambient findability: What we find changes who we become*. O'Reilly Media, 2005.

[128] J. Natt och Dag, V. Gervasi, S. Brinkkemper, and B. Regnell. Speeding up requirements management in a product software company: Linking customer wishes to product requirements through linguistic engineering. In *Proceedings of the 12th International Requirements Engineering Conference*, pages 283–294, 2004.

[129] J. Natt och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karlsson. A feasibility study of automated natural language requirements analysis in market-driven development. *Requirements Engineering*, 7(1):20–33, 2002.

[130] J. Natt och Dag, T. Thelin, and B. Regnell. An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven product development. *Empirical Software Engineering*, 11(2):303–329, 2006.

[131] R. Oliveto. *Traceability management meets information retrieval methods: Strengths and limitations*. PhD thesis, University of Salerno, 2008.

[132] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *International Conference on Program Comprehension*, pages 68–71, 2010.

[133] T. Olsson. *Software information management in requirements and test documentation*. Licentiate thesis, Lund University, 2002.

[134] S. Park, H. Kim, Y. Ko, and J. Seo. Implementation of an efficient requirements analysis supporting system using similarity measure techniques. *Information and Software Technology*, 42(6):429–438, 2000.

[135] A. G Parvathy, B. G Vasudevan, and R. Balakrishnan. A comparative study of document correlation techniques for traceability analysis. In *Proceedings of the 10th International Conference on Enterprise Information Systems, Information Systems Analysis and Specification*, pages 64–69, 2008.

[136] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, pages 71–80, 2008.

[137] K. Pohl, G. Böckle, and F. van der Linden. *Software product line engineering: foundations, principles, and techniques*. Birkhäuser, 2005.

[138] J. Ponte and B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, 1998.

[139] D. Port, A. Nikora, J. Hihn, and L. Huang. Experiences with text mining large collections of unstructured systems development artifacts at JPL. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 701–710, 2011.

[140] J. Randolph. Free-marginal multirater Kappa (multirater K[free]): An alternative to Fleiss' fixed-marginal multirater Kappa. In *Joensuu Learning and Instruction Symposium*, 2005.

[141] S. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.

[142] S. Robertson and J. Robertson. *Mastering the requirements process*. Addison-Wesley Professional, 1999.

[143] S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundation and Trends in Information Retrieval*, 3(4):333–389, 2009.

[144] S. E. Robertson and S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[145] J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, 1971.

[146] P. Runeson, M. Alexandersson, and O. Nyholm. Detection of duplicate defect reports using natural language processing. In *Proceedings of the 29th International Conference on Software Engineering*, pages 499–510, 2007.

[147] P. Runeson, M. Höst, A. Rainer, and B. Regnell. *Case study research in software engineering: Guidelines and examples*. Wiley, 2012.

[148] G. Sabaliauskaite, A. Loconsole, E. Engström, M. Unterkalmsteiner, B. Regnell, P. Runeson, T. Gorschek, and R. Feldt. Challenges in aligning requirements engineering and verification in a large-scale industrial context. In *Requirements Engineering: Foundation for Software Quality*, pages 128–142, 2010.

[149] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[150] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Commununications of the ACM*, 18(11):613–620, 1975.

[151] W. Scacchi. Understanding the requirements for developing open source software systems. *IEEE Software*, 149(1):24–39, 2002.

[152] R. Settimi, J. Cleland-Huang, O. Ben Khadra, J. Mody, W. Lukasik, and C. DePalma. Supporting software evolution through dynamically retrieving traces to UML artifacts. In *Proceedings of the 7th International Workhop on Principles of Software Evolution*, pages 49–54, 2004.

[153] F. Shull, J. Carver, S. Vegas, and N. Juristo. The role of replications in empirical software engineering. *Empirical Software Engineering*, 13(2):211–218, 2008.

[154] A. Singhal. Modern information retrieval: A brief overview. *Data Engineering Bulletin*, 24(2):1–9, 2001.

[155] A. Smeaton and D. Harman. The TREC experiments and their impact on europe. *Journal of Information Science*, 23(2):169–174, 1997.

[156] G. Spanoudakis, A. d'Avila-Garcez, and A. Zisman. Revising rules to capture requirements traceability relations: A machine learning approach. In *Proceedings of the 15th International Conference in Software Engineering and Knowledge Engineering*, 2003.

[157] G. Spanoudakis, A. Zisman, E. Pérez-Miñana, and P. Krause. Rule-based generation of requirements traceability relations. *Journal of Systems and Software*, 72(2):105–127, 2004.

[158] K. Spärck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: Development and comparative experiments. *Information Processing and Management*, 36(6):779–808, 2000.

[159] A. Stone and P. Sawyer. Using pre-requirements tracing to investigate requirements based on tacit knowledge. In *Proceedings of the 1st International Conference on Software and Data Technologies*, pages 139–144, 2006.

[160] H. Sultanov and J. Huffman Hayes. Application of swarm techniques to requirements engineering: Requirements tracing. In *Proceedings of the 18th International Requirements Engineering Conference*, pages 211–220, 2010.

[161] S. Sundaram, J. Huffman Hayes, and A. Dekhtyar. Baselines in requirements tracing. In *Proceedings of the Workshop on Predictor Models in Software Engineering*, pages 1–6, 2005.

[162] S. Sundaram, J. Huffman Hayes, A. Dekhtyar, and A. Holbrook. Assessing traceability of software engineering artifacts. *Requirements Engineering*, 15(3):313–335, 2010.

[163] M. Torchiano and F. Ricca. Impact analysis by means of unstructured knowledge in the context of bug repositories. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, pages 47:1–47:4, 2010.

[164] H. Turtle and B. Croft. Evaluation of an inference network-based retrieval model. *Transactions on Information Systems*, 9(3):187–222, 1991.

[165] B. Van Rompaey and S. Demeyer. Establishing traceability links between unit test cases and units under test. In *Proceedings of the 13th European Conference on Software Maintenance and Reengineering*, pages 209–218, 2009.

[166] E. Voorhees. *TREC: experiment and evaluation in information retrieval*. MIT Press, 2005.

[167] X. Wang, G. Lai, and C. Liu. Recovering relationships between documentation and source code based on the characteristics of software engineering. *Electronic Notes in Theoretical Computer Science*, 243:121–137, 2009.

[168] S. Winkler. Trace retrieval for evolving artifacts. In *Proceedings of the International Workshop on Traceability in Emerging Forms of Software Engineering*, pages 49–56, 2009.

[169] S. Winkler and J. Pilgrim. A survey of traceability in requirements engineering and model-driven development. *Software & Systems Modeling*, 9(4):529–565, 2010.

[170] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: A practical guide*. Springer, 2012.

[171] S. Yadla, J. Huffman Hayes, and A. Dekhtyar. Tracing requirements to defect reports: An application of information retrieval techniques. *Innovations in Systems and Software Engineering*, 1:116–124, 2005.

[172] R. Yin. *Case study research: Design and methods*. Sage Publications, 3rd edition, 2003.

[173] C. Zhai. A brief review of information retrieval models. Technical report, University of Illinois at Urbana-Champaign, 2007.

[174] C. Zhai. Statistical language models for information retrieval: A critical review. *Foundations and Trends in Information Retrieval*, 2(3):137–213, 2008.

[175] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the 10th International Conference on Information and Knowledge Management*, pages 403–410, 2001.

[176] W. Zhao, L. Zhang, Y. Liu, J. Luo, and J. S. Sun. Understanding how the requirements are implemented in source code. In *Proceedings of the Asia-Pacific Software Engineering Conference*, pages 68–77, 2003.

[177] X. Zhou and H. Yu. A clustering-based approach for tracing object-oriented design to requirement. In *Proceedings of the 10th International Conference on Fundamental Approaches to Software Engineering*, pages 412–422, 2007.

[178] X. Zou, R. Settimi, and J. Cleland-Huang. Phrasing in dynamic requirements trace retrieval. In *Proceedings of the International Computer Software and Applications Conference*, pages 265–272, 2006.

[179] X. Zou, R. Settimi, and J. Cleland-Huang. Evaluating the use of project glossaries in automated trace retrieval. In *Proceedings of the International Conference on Software Engineering Research and Practice*, pages 157–163, 2008.

[180] X. Zou, R. Settimi, and J. Cleland-Huang. Improving automated requirements trace retrieval: A study of term-based enhancement methods. *Empirical Software Engineering*, 15(2):119–146, 2010.

PAPER II

INDUSTRIAL COMPARABILITY OF STUDENT ARTIFACTS IN TRACEABILITY RECOVERY RESEARCH - AN EXPLORATORY SURVEY

Abstract

About a hundred studies on traceability recovery have been published in software engineering fora. In roughly half of them, software artifacts developed by students have been used as input. To what extent student artifacts differ from industrial counterparts has not been fully explored in the literature. We conducted a survey among authors of studies on traceability recovery, including both academics and practitioners, to explore their perspectives on the matter. Our results indicate that a majority of authors consider software artifacts originating from student projects to be only partly representative to industrial artifacts. Moreover, only few respondents validated student artifacts for industrial representativeness. Furthermore, our respondents made suggestions for improving the description of artifact sets used in studies by adding contextual, domain-specific and artifact-centric information. Example suggestions include adding descriptions of processes used for artifact development, meaning of traceability links, and the structure of artifacts. Our findings call for further research on characterization and validation of software artifacts to support aggregation of results from empirical studies.

Markus Borg, Krzysztof Wnuk, and Dietmar Pfahl, *In Proceedings of the 16th European Conference on Software Maintenance and Reengineering*, 2012

1 Introduction

To advance both the state-of-art and state-of-practice in software engineering, empirical studies (i.e., surveys, experiments, case studies) have to be conducted [7, 20]. In order to investigate the cause-effect relationships of introducing new methods, techniques or tools in software engineering, controlled experiments are commonly used as the research method. Despite the benefits resulting from the controlled environment that can be created in this fixed research design [19, 24], controlled experiments are expensive due to the involvement of human subjects. Therefore, controlled experiments are often conducted with university students - and not with engineers working in industry.

Several researchers have studied the differences between using students and practitioners as subjects in software engineering studies, since the inadequate choice of subjects might be a considerable threat to the validity of the study results [8]. A number of publications report that the differences are only minor, thus using students is reasonable under certain circumstances [2, 8, 9, 14, 23]. Senior students, representing the next generation of professional software engineers, are relatively close to the population of interest in studies aiming at emulating professional behavior [14]. Further, for relatively small tasks, trained students have been shown to perform comparably to practitioners in industry [8, 23].

However, the question whether software artifacts (referred to as only 'artifacts' in this paper) produced by students should be used in empirical studies has been less explored. How useful are results from such studies when it comes to generalizability to industrial contexts? Using student artifacts is often motivated by low availability of industrial artifacts due to confidentiality issues. A qualification of the validity of student artifacts is particularly important for the domain of traceability recovery, since student artifacts frequently have been used for tool evaluations [5, 6, 18]. Since several software maintenance tasks (such as change impact analysis and regression testing) depend on up-to-date traceability information [12], it is fundamental to understand the nature of experimental artifact sets.

Furthermore, as presented in more detail in Section II, the reported characterization of artifact sets used as input to experiments on traceability recovery is typically insufficient. According to Jedlitschka *et al.*, inadequate reporting of empirical research commonly impedes integration of study results into a common body of knowledge [13]. This applies also to traceability recovery research. First, insufficient reporting makes it harder to assess the validity of results using student artifacts (even if artifacts have been made available elsewhere). Second, it hinders aggregation of empirical results, particularly when closed industrial artifact sets have been used (that never can be made available).

In this paper, we present a study exploring differences between Natural Language (NL) artifacts originating from students and practitioners. We conducted a questionnaire-based survey of researchers with experience in doing experiments on traceability recovery using Information Retrieval (IR) approaches. The survey

builds upon results from a literature study, which will be published in full detail elsewhere.

This paper is structured as follows. Section 2 presents background, including a short overview of related work on IR-based traceability recovery and using students in software engineering experiments. Section 3 presents the research design and how the survey was conducted. In Section 4 we present and analyze our results in comparison to the literature. Section 5 describes threats to validity. Finally, Section 6 concludes the paper and discusses future work.

2 Background and Related Work

Using students as subjects in empirical software engineering studies has been considered as reasonable by several researchers [2, 8, 9, 14, 23]. Kitchenham *et al.* claim that students are the next generation of software professionals and that they are relatively close to the population of interest (practitioners) [14]. Kuzniarz *et al.* concluded that students are good subjects under certain circumstances and proposes a classification of the possible types of students used in an experiment [15]. Svahnberg *et al.* investigated if students understand the way how industry acts in the context of requirements selection [23].

Höst *et al.* investigated the incentives and experience of subjects in experiments and proposed a classification scheme in relation to the outcome of an experiment [9]. Although Höst *et al.* distinguished between artificial artifacts (such as produced by students during a course) and industrial artifacts as part of the incentives in the proposed classification, guidelines on how to assess the two types of artifacts are not provided in this work. Moreover, none of the mentioned studies investigate whether artifacts produced by students are comparable to artifacts produced in industry and how possible discrepancies could be assessed.

Several empirical studies on traceability recovery were conducted using student subjects working on artifacts originating from student projects. De Lucia *et al.* evaluated the usefulness of supported traceability recovery in a study with 150 students in 17 software development projects at the University of Salerno [6]. They also conducted a controlled experiment with 32 students using student artifacts [5]. Natt och Dag *et al.* conducted another controlled experiment in an academic setting, where 45 students were solving tracing tasks on artifacts produced by students [18].

During the last decade, several researchers proposed expressing the traceability challenge, i.e., identifying related artifacts, as an IR problem [1, 5, 17]. The approach suggests traceability links based on textual similarity between artifacts, since text in NL is the common format of information representation in software development [17]. The underlying assumption is that developers use similar language when referring to the same functionality across artifacts.

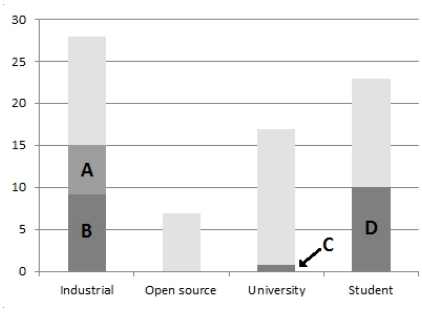


Figure 1: Origin of artifacts used in IR-based traceability recovery evaluations. Artifact sets in darker grey are available at COEST (A=MODIS, B=CM-1, C=Waterloo, D=EasyClinic).

	Artifact set	Artifact types	Origin / Domain	Size (#artifacts)
A	MODIS	Requirements	NASA / Embedded	48
B	CM-1	Requirements	NASA / Embedded	555
C	Waterloo	Requirements	23 student projects	23 x ~75
D	EasyClinic	Requirements code, test cases	Student project	160

Table 1: Publicly available artifact sets at COEST (Oct 23, 2011).

In an ongoing literature review, we have identified 59 publications on IR-based traceability recovery of NL artifacts. Figure 1 shows the reported origin of artifacts used in evaluations of traceability recovery tools, classified as industrial artifacts, open source artifacts, university (artifacts developed in university projects, role of developers unspecified) or student (deliverables from student projects). Several publications use artifacts from more than one category, some do not report the origin of the artifacts used for evaluations. As Figure 1 shows, a majority of the artifacts originate from an academic environment, i.e. they have been developed in university or student projects.

The Center of Excellence for Software Traceability (COEST) has collected and published four artifact sets (see Table 1), that constitute the de-facto benchmarks for IR-based traceability recovery research. In Figure 1, darker grey color represents artifact sets available at COEST. Among the 59 identified publications, the most frequently used artifact sets in traceability recovery studies are EasyClinic (marked with a letter D, 10 publications), CM-1 (B, 9 publications) and MODIS (A, 6 publications).

In 2005, Huffman Hayes and Dekhtyar proposed "A framework for comparing requirements tracing experiments" [10]. The framework focuses on developing, conducting and analyzing experiments, but also suggests information about artifacts and contexts that are worth reporting. They specifically say that the average size of an artifact is of interest, but that it rarely is specified in research papers. Furthermore, they propose characterizing the quality of the artifacts and the importance of both the domain and object of study (on a scale from convenience to safety-critical).

Moreover, even though the framework was published in 2005, our review of the literature revealed that artifact sets often are presented in rudimentary fashion in the surveyed papers. The most common way to characterize artifact sets in the surveyed papers is to report its origins together with a brief description of the functionality of the related system, its size and the types of artifacts included. Size is reported as the number of artifacts and the number of traceability links between them. This style of reporting was applied in 49 of the 59 publications (83%). Only three publications thoroughly describe the context and process used when the artifacts were developed. For example, Lormans *et al.* well describe the context of their case study at LogicaCMG [16].

Apart from mentioning size and number of links, some publications present more detail regarding the artifacts. Six publications report descriptive statistics of individual artifacts, including average size and number of words. Being even more detailed, Huffman Hayes *et al.* reported two readability measures to characterize artifact sets, namely Flesch Reading Ease and Flesch-Kincaid Grade Level [11]. Another approach was proposed by De Lucia *et al.* [5]. They reported subjectively assessed quality of different artifact types, in addition to the typical size measures. As stressed by Jedlitschka *et al.* proper reporting of traceability recovery studies is important, since inadequate reporting of empirical research commonly impedes integration of study results into a common body of knowledge [13].

3 Research Design

This section presents the research questions, the research methodology, and the data collection procedures used in our study. The study is an exploratory follow-up to the ongoing literature review mentioned in Section 2. Table 5 presents the research questions governing this study. The research questions investigate whether the artifacts used in the reported studies are considered comparable to their industrial counterparts by our respondents. Moreover, the questions aim at exploring how to support assessing the comparability by augmenting the descriptions of the used artifacts.

For this study, we chose a questionnaire-based survey as the tool to collect empirical data, since it helps reaching a large number of respondents from geographically diverse locations [21]. Also, a survey provides flexibility and is convenient

	Research question	Aim	Example answer
RQ1	When used as experiment inputs, how comparable are artifacts produced by students to their industrial counterparts?	Understand to what degree respondents, both in academia and industry, consider industrial and student artifacts to be comparable.	"As a rule, the educational artifacts are simpler."
RQ2	How are artifacts validated before being used as input to experiments?	Survey if and how student artifacts are validated before experiments are conducted.	"Our validation was based on expert opinion."
RQ3	Is the typically reported characterization of artifact sets sufficient?	Do respondents, both in academia and industry, consider that the way natural language artifacts are described is good enough.	"I would argue that it should also be characterized by the process by it was developed."
RQ4	How could artifacts be described to better support aggregation of empirical results?	Explore whether there are ways to improve the way natural language artifacts are presented.	"The artifacts should be combined with a task that is of principal cognitive nature."
RQ5	How could the difference between artifacts originating from industrial and student projects be measured?	Investigate if there are any measures that would be particularly suitable to compare industrial and student artifacts.	"The main difference is the verbosity."

Table 2: Research questions of the study. All questions are related to the context of traceability recovery studies.

to both researchers and participants [7]. The details in relation to survey design and data collection are outlined in the section that follows.

3.1 Survey design

Since the review of literature resulted in a substantial body of knowledge on IR-based approaches to traceability recovery, we decided to use the authors of the identified publications as our sample population. Other methods to recover traceability have been proposed, including data mining [22] and ontology-based recovery [25], however the majority of traceability recovery publications apply IR techniques. Furthermore, it is well-known that IR is sensitive to the input data used in evaluations [4].

The primary aim of this study was to explore researchers' views on the comparability between NL artifacts produced by students and practitioners. We restricted the sample to authors with documented experience, i.e., published peer-reviewed research articles, of using either student or industrial artifact sets in IR-based traceability recovery studies. Consequently, we left out authors who exclusively used artifacts from the open source domain.

The questionnaire was constructed through a brainstorming session with the authors, using the literature review as input. To adapt the questions to the respondents regarding the origin of the artifacts used, three versions of the questionnaire were created:

- **STUD**. A version for authors of published studies on traceability recovery using student artifacts. This version was most comprehensive since it contained more questions. Thus it was sent to authors, if at least one publication using student artifacts had been identified.

- **UNIV**. A version for authors using artifacts originating from university projects. This version included a clarifying question on whether the artifacts were developed by students or not, followed by the same detailed questions about student artifacts as in version STUD. This question was used to filter out answers related to student artifacts.

- **IND**. A subset of STUD, sent to authors who only had published traceability recovery studies using industrial artifacts.

We piloted the questionnaire using five senior software engineering researchers, including a native English speaker. The three versions of the questionnaire were then refined, the final versions are presented in the Appendix. The mapping between research questions and questions in the questionnaire is presented in Table 3.

Research questions	Questionnaire questions
RQ1	QQ1, *QQ4*, QQ6
RQ2	*QQ4*, QQ5
RQ3	QQ2
RQ4	QQ3
RQ5	*QQ4*, QQ7

Table 3: Mapping between research questions and the questionnaire. QQ4 was used as a filter.

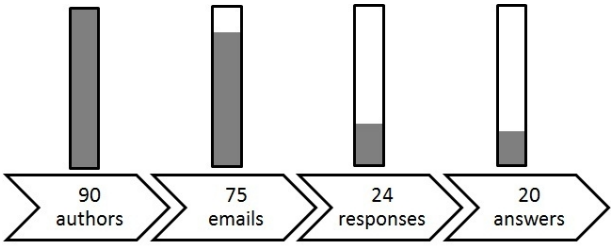


Figure 2: Survey response rate.

3.2 Survey execution and analysis

The questionnaires were distributed via email, sent to the set of authors described in Section 3.1. As Figure 2 depicts, in total 90 authors were identified. We were able to send emails that appeared to reach 75 (83%) of them. Several mails returned with no found recipient and in some cases no contact information was available. In those few cases we tried contacting current colleagues; nevertheless there remained 15 authors (17%) we did not manage to send emails successfully. The mails were sent between September 27 and October 12, 2011. After one week, reminders were sent to respondents who had not yet answered the survey.

24 authors (32%) responded to our emails; however four responses did not contain answers to the survey questions. Among them, two academics referred to other colleagues more suitable to answer the questionnaire (all however already included in our sample) and two practitioners claimed to be too disconnected from research to be able to answer with a reasonable effort. Thus, the final set of complete answers included 20 returned questionnaires. This yielded a response rate of 27%.

The survey answers were analyzed by descriptive statistics and qualitative categorization of the answers. The results and the analysis are presented in Section 4.

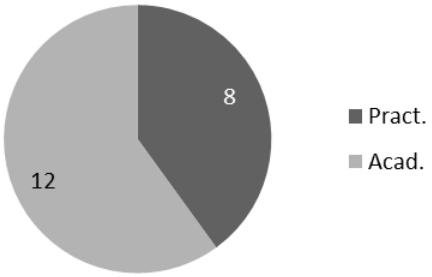


Figure 3: Current main affiliation of respondents.

	STUD	UNIV	IND
Academics	8	2	2
Practitioners	3	0	5
Total	11	2	7

Table 4: Mapping between research questions and the questionnaire. QQ4 was used as a filter.

4 Results and Analysis

In this section, the results from the survey of authors are presented and discussed.

4.1 Demographics and sample characterization

For the 20 authors of publications on IR-based traceability recovery who answered the questionnaire, Figure 3 depicts the distribution of practitioners and academics based on current main affiliation. 40% of the respondents are currently working in industry. Our respondents represent all combinations of academics and practitioners from Europe, North America and Asia. Table 4 presents how many answers we received per questionnaire version. Both respondents answering UNIV reported that students had developed the artifacts in their university projects (QQ4), thus at least twelve of the respondents had experience with student artifacts in traceability recovery experiments.

4.2 Representativeness of student artifacts (RQ1)

In this subsection, we present the view of our respondents on the representativeness of software artifacts produced by students. In this case, we investigated if our respondents agree with the statement that software artifacts produced by students are representative of software artifacts produced in industry (see QQ1, and QQ6 filtered by QQ4 in the Appendix). QQ6 overlaps QQ1 by targeting specific

publications. To preserve the anonymity of the respondents, the analyses of the questions are reported together.

Figure 4 shows survey answers to the statement "Software artifacts produced by students (used as input in traceability experiments) are representative of software artifacts produced in industry" (QQ1). Black color represents answers from practitioners, grey color answers from academics. Half of the respondents fully or partly disagree to the statement. Academics answered this question with a higher degree of consensus than practitioners. No respondent totally agreed to the statement.

Several respondents decided to comment on the comparability of student artifacts. Two of them, both practitioners answering QQ1 with '4', pointed out that trained students actually might produce NL artifacts of higher quality than engineers in industry. One of them clarified: "In industry, there are a lot of untrained 'professionals' who, due to many reasons including time constraints, produce 'flawed' artifacts". Another respondent answered QQ1 with '2', but stressed that certain student projects could be comparable to industrial counterparts, for instance in the domain of web applications. On the other hand, he explained, would they not at all be comparable for domains with many process requirements such as finance and aviation. Finally, one respondent mentioned the wider diversity of industrial artifacts, compared to artifacts produced by students: "I've seen ridiculously short requirements in industry (5 words only) and very long ones of multiple paragraphs. Students would be unlikely to create such monstrous requirements!" and also added "Student datasets are MUCH MUCH smaller (perhaps 50-100 artifacts compared to several thousands)".

Three of our respondents mentioned the importance of understanding the incentives of the developers of the artifacts. This result confirms the findings by Höst *et al.* [9]. The scope and lifetime of student artifacts are likely to be much different for industrial counterparts. Another respondent (academic) supported this claim and also stressed the importance of the development context: "The vast majority [of student artifacts] are developed for pedagogical reasons - not for practical reasons. That is, the objective is not to build production code, but to teach students." According to one respondent (practitioner), both incentives and development contexts are playing an important role also in industry: "Industrial artifacts are created and evolved in a tension between regulations, pressing schedule and lacking motivation /—/ some artifacts are created because mandated by regulations, but no one ever reads them again, other artifacts are part of contracts and are, therefore, carefully formulated and looked through by company lawyers etc."

These results are not surprising, and lead to the conclusion that NL artifacts produced by students are understood to be less complex than their industrial counterparts. However, put in the light of related work outlined in Section 2, the results can lead to interesting interpretations. As presented in Figure 1, experiments on traceability recovery frequently use artifacts developed by students as input. Also, as presented in Table 1, two of four publicly available artifact sets at COEST orig-

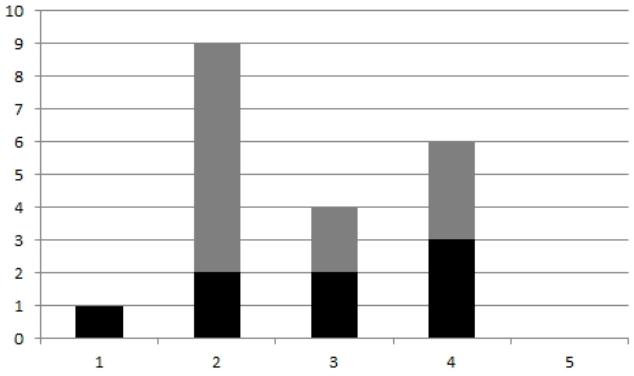


Figure 4: Are student artifacts representative to industrial counterparts? (1 = totally disagree, 5 = totally agree) (QQ1)

inate from student projects. Nevertheless, our respondents mostly disagreed that these artifacts are representative of NL artifacts produced in industry.

4.3 Validation of experimental artifacts (RQ2)

In this subsection, we present the results from QQ5 which is related to research question RQ2. QQ5, filtered by QQ4, investigates whether student artifacts, when used in traceability recovery experiments, were validated for industrial representativeness.

We received answers to QQ5 from 13 respondents (questionnaire versions STUD and UNIV). The distribution of answers is depicted in Figure 4. Among the five respondents who validated student artifacts being used as experimental input, three respondents focused on robustness of the experiment output (of the experiment in which the artifacts were used as input). The robustness was assessed by comparing experimental results to experiments using industrial artifacts. As another approach to validation, two respondents primarily used expert opinion to evaluate the industrial comparability of the student artifacts. Finally, three respondents answered that they did not conduct any explicit validation of the industrial comparability at all.

Neither answering ‘yes’ nor ‘no’ to QQ5, five respondents discussed the question in more general terms. Two of them stressed the importance of conducting traceability recovery experiments using realistic tasks. One respondent considered it significant to identify in which industrial scenario the student artifacts would be representative and said “The same student artifacts can be very ‘industrial’ if we think at a hi-tech startup company, and totally ‘unindustrial’ if we think at Boe-

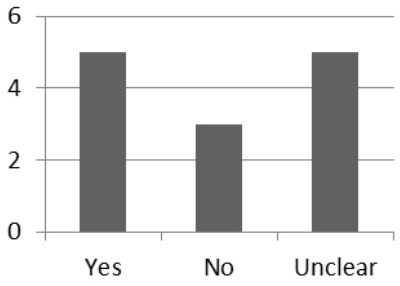


Figure 5: Were the student artifacts validated for industrial comparability? (QQ5)

ing". Another respondent claimed that they had focused on creating an as general tracing task as possible.

Only a minority of researchers who used student artifacts to evaluate IR-based traceability recovery explicitly answered with 'yes' to this question, suggesting that it is no widespread common practice. Considering the questionable comparability of artifacts produced by students, confirmed by QQ1, this finding is remarkable. Simply assuming that there is an industrial context where the artifacts would be representative might not be enough. The validation that actually takes place appears to be ad-hoc, thus some form of supporting guidelines would be helpful.

4.4 Adequacy of artifact characterization (RQ3)

In this subsection, we present the results from asking our respondents whether the typical way of characterizing artifacts used in experiments (mainly size and number of correct traceability links) is sufficient. In Figure 6, we present answers to QQ2 which is related to RQ3. Black color represents practitioners, grey color academics. Two respondents (both academics) considered this to be a fully sufficient characterization. The distribution of the rest of the answers, both for practitioners and academics, shows mixed opinions.

Respondents answering with '1' (totally insufficient) to QQ2 motivated their answers by claiming: simple link existence being too rudimentary, complexity of artifact sets must be presented and the meaning of traceability links should be clarified. On the other hand, seven respondents answered with '4' or '5' (5=fully sufficient). Their motivations included: tracing effort is most importantly proportional to the size of the artifact set and experiments based on textual similarities are reliable. However, two respondents answering with '4' also stated that information density and language are important factors and that the properties of the traceability links should not be missed.

More than 50% of all answers to QQ2 were marking options '1', '2' or '3'. Thus a majority of the respondents answering this question either disagree with

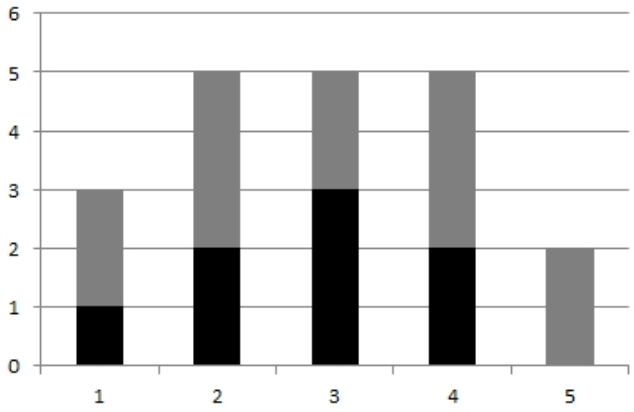


Figure 6: Is size and traceability link number sufficient to characterize an artifact set? (1 = totally insufficient, 5 = fully sufficient) (QQ2)

the question statement or have a neutral opinion. This result is contrasting with published literature, in which we found that characterization of input artifacts in traceability experiments is generally brief (see Section 2). There may be two possible explanations of this misalignment. Either the authors don't see the need of providing more descriptions of the used artifact sets (this may be the remaining minority of the answers), or the complementary metrics and important characterization factors are unknown. We believe that the result supports the second explanation as only limited work including explicit guidance has been published to date. Two respondents answered with '4' without motivating their choices. To conclude, since our review of literature found that the characterization of input artifacts in traceability experiments is generally brief (see Section 2), this result justifies our research efforts and calls for further explanatory research.

Our results indicate that authors are aware that there are other significant features of artifact sets than the typically reported size and total number of links (see also results in Section IV.E). Apparently, there seems to be a gap between what is considered a preferred characterization and what actually is reported in publications. The gap could have been partly mitigated if the research community to a higher extent had accepted "A framework for requirements tracing experiments", since it partly covers artifact set descriptions [10]. However, the results also indicate that parts of the research community think that the basic reporting is a good start.

4.5 Improved characterization (RQ4)

In this section we provide results for the RQ4, exploring ways to improve the way NL artifacts are reported, addressed by QQ3. Eleven respondents, six academics and five practitioners, suggested explicit enhancements to artifact set characterization, other respondents answered more vaguely. Those suggestions are collected and organized below into the three classes Contextual (describes the environment of the artifact development), Link-related (describes properties of traceability links) and Artifact-centric (describes the artifacts). In total, 23 aspects to additionally characterize artifacts were suggested.

Contextual aspects:

- Domain from which the artifacts originate
- Process used when artifact was developed (agile/spiral/waterfall etc., versioning, safety regulations)
- When in the product lifecycle the artifacts were developed
- Maturity/evolution of the artifacts (years in operation, #reviews, #updates)
- Role and personality of the developer of the artifacts
- Number of stakeholders/users of the artifacts
- Tasks that are related to the artifact set

Link-related aspects:

- Meaning of a traceability link (depends on, satisfies, based on, verifies etc.)
- Typical usage of traceability links
- Values and costs of traceability links (Value of correct link, cost of establishing link, establishing false link, missing link)
- Person who created the golden standard of links (practitioners, researchers, students, and their incentives)
- Quality of the golden standard of traceability links
- Link density
- Distribution of inbound/outbound links

Artifact-centric aspects:

- Size of individual artifacts
- Language (Natural, formal)
- Complexity of artifacts
- Verbosity of artifacts
- Artifact redundancy/overlap
- Artifact granularity (Full document/chapter/page/ section etc.)
- Quality/maturity of artifact (#defects reported, draft/reviewed/released)
- Structure/format of artifact (structured/semi-structured/unstructured information)
- Information density

As our survey shows, several authors have ideas about additional artifact set features that would be meaningful to report. Thus most authors both are of the opinion that artifact sets should be better characterized, and also have suggestions for how it could be done. Still, despite also being stressed in Huffman Hayes and Dekhtyars framework from 2005, it has not reached the publications. However, we collected many requests for "what" to describe, but little input on the "how" (i.e. 'what' = state complexity / 'how' = how to measure complexity?). This discrepancy can be partly responsible for the insufficient artifact set characterizations. A collection of how different aspects might be measured, tailored for reporting artifact sets used in traceability recovery studies, appears to be a desirable composition.

One might argue that several of the suggested aspects are not applicable to student projects. This is in line with both what Höst *et al.* [9] and our respondents stated, purpose and lifecycle of student artifacts are rarely representative for industrial settings. Thus, aspects such as maturity, evolution and stakeholders usually are unfeasible to measure. Again, this indicates that artifacts originating from student projects might be too trivial, resulting in little more empirical evidence than proofs-of-concept.

4.6 Measuring student/industrial artifacts (RQ5)

In this section, we present results in relation to RQ5, concerning the respondents' opinions about how differences between NL artifacts developed by students and industrial practitioners can be assessed. QQ7, filtered by QQ4, provides answers to this question.

A majority of the respondents of STUD and UNIV commented on the challenge of measuring differences between artifacts originating from industrial and student projects. Only four respondents explicitly mentioned suitable aspects to investigate. Two of them suggested looking for differences in quality, such as maintainability, extensibility and ambiguities. One respondent stated that the main differences are related to complexity (students use more trivial terminology). On the other hand, one academic respondent instead claimed that "In fact artifact written by students are undoubtedly the most verbose and better argued since their evaluation certainly depends on the quality of the documentation". Yet another respondent, a practitioner, answered that the differences are minor.

Notably, one respondent to QQ7 warned about trying to measure differences among artifacts, motivated by the great diversity in industry. According to the respondent, there is no such thing as an average artifact. "What is commonly called 'requirements' in industry can easily be a 1-page business plan or a 15-volumes requirements specification of the International Space Station", the respondent explained.

To summarize, the results achieved for QQ7 confirm our expectations that measuring the comparability is indeed a challenging task. Obviously, there is no simple measure to aim for. This is also supported by QQ5, the few validations of student artifacts that the respondents reported utilized only expert opinion or replications with industrial artifacts.

5 Threats to Validity

This section provides a discussion of the threats to validity in relation to research design and data collection phases as well as in relation to results from the study. The discussion of the threats to validity is based on the classification proposed by Wohlin *et al.* [24], focusing on threats to construct, internal and external validity.

Construct validity is concerned with the relation between the observations during the study and the theories in which the research is grounded. The exact formulations of the questions in the used questionnaire are crucial in survey research as misunderstanding or misinterpreting the questions can happen. We alleviated this threat to construct validity by revising the questionnaire by an independent reviewer (except the authors of the paper who also revised the questions) who is a native English speaker and writer. To further minimize threats to construct validity, a pilot study was conducted on five senior researchers in software engineering. Still, the subjectivity of the data provided by our respondents can negatively influence the interpretability of the results. Due to a relatively low number of data points, the mono-operational bias threat to construct validity is not fully addressed. The anonymity of respondents was not guaranteed as the survey was sent via email; this leaves the evaluation apprehension threat unaddressed. Since the research is exploratory, the experimenter expectancies threat to construct validity

is minimized. Finally, the literature survey conducted as the first step of the study helps to address the mono-method threat to construct validity, which however still requires further research to fully alleviate it.

Internal validity concerns confounding factors that can affect the causal relationship between the treatment and the outcome. By performing the review of the questionnaire questions, the instrumentation threat to internal validity was addressed. On the other hand, the selection bias can still threaten the internal validity as the respondents were not randomly selected. We have measured the time needed to answer the survey in the pilot study; therefore the maturation threat to internal validity is alleviated. Finally, the selection threat to internal validity should be mentioned here since respondents of the survey were volunteers who, according to Wohlin *et al.*, are not representative for the whole populations [24].

External validity concerns the ability to generalize the results of the study to industrial practice. We have selected a survey research method in order to target more potential respondents from various countries, companies and research groups and possibly generate more results [7]. Still, the received number of responses is low and thus not a strong basis for extensive generalizations of our findings. However, the external validity of the results achieved is acceptable when considering the exploratory nature of this study.

6 Discussion and Concluding Remarks

We have conducted an exploratory survey of the comparability of artifacts used in IR-based traceability recovery experiments, originating from industrial and student projects. Our sample of authors of related publications confirms that artifacts developed by students are only partially comparable to industrial counterparts. Nevertheless, it commonly happens that student artifacts used as input to experimental research are not validated with regards to their industrial representativeness.

Our results show that, typically, artifact sets are only rudimentarily described, despite the experimental framework proposed by Huffman Hayes and Dekhtyar in 2005. We found that a majority of authors of traceability recovery publications think that artifact sets are inadequately characterized. Interestingly, a majority of the authors explicitly suggested features of artifact sets they would prefer to see reported. Suggestions include general aspects such as contextual information during artifact development and artifact-centric measures. Also, domain-specific (link-related) aspects were proposed, specifically applicable to traceability recovery.

The explanatory part of this study, should be followed by an in-depth study validating the proposals made by the respondents and aim at making the proposals more operational. This in turn could lead to characterization schemes that help assess the generalizability of study results using student artifacts. The results could complement Huffman Hayes and Dekhtyars framework [10] or be used as an empirical foundation of a future revision. Moreover, studies similar to this one should

Experimental setup	Practitioners as subjects	Students as subjects
Industrial artifacts	Too expensive	Students do not understand
Student artifacts	Not explored	Uncertain generalizability

Figure 7: Risks involved in different combinations of subjects and artifacts in traceability recovery studies.

be conducted for other application domains where student artifacts frequently are used as input to experimental software engineering, such as regression testing, cost estimation and model-driven development.

Clearly, researchers need to be careful when designing traceability recovery studies. Previous research has shown that using students as experimental subjects is reasonable [2, 8, 9, 14, 23]. However, according to our survey, the validity of using student artifacts is uncertain. Unfortunately, industrial artifacts are hard to get access to. Furthermore, even with access to industrial artifacts, researchers might not be permitted to show them to students. And even with that permission, students might lack the domain knowledge necessary to be able to work with them. Figure 7 summarizes general risks involved in different combinations of subjects and artifacts in traceability recovery studies. The most realistic option, conducting studies on practitioners working with industrial artifacts, is unfortunately often hard to accomplish with a large enough number of subjects. Instead, several previous studies used students solving tasks involving industrial artifacts [3, 12] or artifacts developed in student projects [5, 6, 18]. However, these two experimental setups introduce threats either related to construct validity or external validity. The last option, conducting studies with practitioners working with student artifacts, has not been attempted. We plan to further explore the possible combinations in future work.

Acknowledgement

Thanks go to the respondents of the survey. This work was funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering[1]. Special thanks go to David Callele for excellent language-related comments.

[1] http://ease.cs.lth.se

Appendix

	Questionnaire	Used in versions
QQ1	Would you agree with the statement: "Software artifacts produced by students (used as input in traceability experiments) are representative of software artifacts produced in industry?" (Please select one number. 1 = totally disagree, 5 = totally agree) 1—2—3—4—5	STUD / UNIV / IND
QQ2	Typically, datasets containing software artifacts used as input to traceability experiments are characterized by size and number of correct traceability links. Do you consider this characterization as sufficient? Please explain why you hold this opinion. (Please select one number. 1 = totally disagree, 5 = totally agree) 1—2—3—4—5	STUD / UNIV / IND
QQ3	What would be a desirable characterization of software artifacts to enable comparison (for example between software artifacts developed by students and industrial practitioners)?	STUD / UNIV / IND
QQ4	In your experiment, you used software artifacts developed in the university project [NAME OF PROJECT]. Were the software artifacts developed by students?	UNIV
QQ5	Did you evaluate whether the software artifacts used in your study were representative of industrial artifacts? If you did, how did you perform this evaluation?	STUD / UNIV
QQ6	How representative were the software artifacts you used in your experiment of industrial software artifacts? What was the same? What was different?	STUD / UNIV
QQ7	How would you measure the difference between software artifacts developed by students and software artifacts developed by industrial practitioners?	STUD / UNIV

Table 5: Research questions of the study. All questions are related to the context of traceability recovery studies.

Bibliography

[1] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering traceability links between code and documentation. In *Transactions on Software Engineering*, volume 28, pages 970–983, 2002.

[2] P. Berander. Using students as subjects in requirements prioritization. In *Prodeedings of the International Symposium on Empirical Software Engineering*, pages 167–176, August 2004.

[3] M. Borg and D. Pfahl. Do better IR tools improve the accuracy of engineers' traceability recovery? In *Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering*, pages 27–34, 2011.

[4] C. Borgman. *From Gutenberg to the global information infrastructure: Access to information in the networked world*. MIT Press, 2003.

[5] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Recovering traceability links in software artifact management systems using information retrieval methods. *Transactions on Software Engineering and Methodology*, 16(4), 2007.

[6] A. De Lucia, R. Oliveto, and G. Tortora. Assessing IR-based traceability recovery tools through controlled experiments. *Empirical Software Engineering*, 14(1):57–92, 2009.

[7] S. Easterbrook, J. Singer, M. Storey, and D. Damian. Selecting empirical methods for software engineering research. In F. Shull, J. Singer, and D. Sjöberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer, 2008.

[8] M Höst, B. Regnell, and C. Wohlin. Using students as subjects: A comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, 5(3):201–214, 2000.

[9] M. Höst, C. Wohlin, and T. Thelin. Experimental context classification: Incentives and experience of subjects. In *Proceedings of the 27th international conference on Software engineering*, pages 470–478, 2005.

[10] J. Huffman Hayes and A. Dekhtyar. A framework for comparing requirements tracing experiments. *Interational Journal of Software Engineering and Knowledge Engineering*, 15(5):751–781, 2005.

[11] J. Huffman Hayes, A. Dekhtyar, and S. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *Transactions on Software Engineering*, 32(1):4–19, 2006.

[12] J. Huffman Hayes, A. Dekhtyar, S. Sundaram, A. Holbrook, S. Vadlamudi, and A. April. REquirements TRacing on target (RETRO): improving software maintenance through traceability recovery. *Innovations in Systems and Software Engineering*, 3(3):193–202, 2007.

[13] A. Jedlitschka, M. Ciolkowski, and D. Pfahl. Reporting experiments in software engineering. In F. Shull, J. Singer, and D. Sjöberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 201–228. Springer, London, 2008.

[14] B. Kitchenham, S. Pfleeger, L Pickard, P. Jones, D. Hoaglin, K. El Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *Transactions on Software Engineering and Methodology*, 28(8):721–734, 2002.

[15] L. Kuzniarz, M. Staron, and C. Wohlin. Students as study subjects in software engineering experimentation. In *Proceedings of the 3rd Conference on Software Engineering Research and Practise in Sweden*, 2003.

[16] M. Lormans, H-G. Gross, A. van Deursen, R. van Solingen, and A. Stehouwer. Monitoring requirements coverage using reconstructed views: An industrial case study. In *Procedings of the 13th Working Conference on Reverse Engineering*, pages 275–284, 2006.

[17] A. Marcus and J. Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. In *Proceedings of the International Conference on Software Engineering*, pages 125–135, 2003.

[18] J. Natt och Dag, T. Thelin, and B. Regnell. An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven product development. *Empirical Software Engineering*, 11(2):303–329, 2006.

[19] B. Robson. *Real world research*. Blackwell, 2nd edition, 2002.

[20] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

[21] J. Singer, S. Sim, and T. Lethbridge. Software engineering data collection for field studies. In F. Shull, J. Singer, and D. Sjöberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 9–34. Springer, 2008.

[22] G. Spanoudakis, A. d'Avila-Garcez, and A. Zisman. Revising rules to capture requirements traceability relations: A machine learning approach. In *Proceedings of the 15th International Conference in Software Engineering and Knowledge Engineering*, 2003.

[23] M. Svahnberg, A. Aurum, and C. Wohlin. Using students as subjects: An empirical evaluation. In *Proceedings of the 2nd International Symposium on Empirical Software Engineering and Measurement*, pages 288–290, 2008.

[24] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: An introduction*. Kluwer Academic Publications, 1st edition, 1999.

[25] Y. Zhang, R. Witte, J. Rilling, and V. Haarslev. Ontological approach for the semantic recovery of traceability links between software artefacts. *IET Software*, 2(3):185–203, 2008.

PAPER III

EVALUATION OF TRACEABILITY RECOVERY IN CONTEXT: A TAXONOMY FOR INFORMATION RETRIEVAL TOOLS

Abstract

Background: Development of complex, software intensive systems generates large amounts of information. Several researchers have developed tools implementing information retrieval (IR) approaches to suggest traceability links among artifacts. *Aim:* We explore the consequences of the fact that a majority of the evaluations of such tools have been focused on benchmarking of mere tool output. *Method:* To illustrate this issue, we have adapted a framework of general IR evaluations to a context taxonomy specifically for IR-based traceability recovery. Furthermore, we evaluate a previously proposed experimental framework by conducting a study using two publicly available tools on two datasets originating from development of embedded software systems. *Results:* Our study shows that even though both datasets contain software artifacts from embedded development, the characteristics of the two datasets differ considerably, and consequently the traceability outcomes. *Conclusions:* To enable replications and secondary studies, we suggest that datasets should be thoroughly characterized in future studies on traceability recovery, especially when they can not be disclosed. Also, while we conclude that the experimental framework provides useful support, we argue that our proposed context taxonomy is a useful complement. Finally, we discuss how empirical evidence of the feasibility of IR-based traceability recovery can be strengthened in future research.

Markus Borg, Per Runeson, and Lina Brodén, *In Proceedings of the 16th International Conference on Evaluation and Assessment in Software Engineering*, 2012

1 Introduction

Large-scale software development generates large amounts of information. Enabling software engineers to efficiently navigate the document space of the development project is crucial. One typical way to structure the information within the software engineering industry is to maintain traceability, defined as "the ability to describe and follow the life of a requirement, in both a forward and backward direction" [18]. This is widely recognized as an important factor for efficient software engineering as it supports activities such as verification, change impact analysis, program comprehension, and software reuse [2].

Several researchers have proposed using information retrieval (IR) techniques to support maintenance of traceability information [10, 23, 28, 29]. Traceability can be maintained between any software artifacts, i.e., any piece of information produced and maintained during software development. Textual content in natural language (NL) is the common form of information representation in software engineering [31]. Tools implementing IR-based traceability recovery suggest traceability links based on textual similarities, for example between requirements and test cases. However, about 50% of the evaluations have been conducted using one of the four datasets available at the *Center of Excellence for Software Traceability* (COEST)[1] [6]. Consequently, especially the *CM-1* and *EasyClinic* datasets have turned into de-facto benchmarks of IR-based traceability recovery, further amplified by "traceability challenges" issued by the *Traceability in Emerging Forms of Software Engineering* (TEFSE) workshop series.

Developing a repository of benchmarks for traceability research is a central part of COEST's vision, and thus has been discussed in several publications [4, 7, 12, 13]. As another contribution supporting benchmarking efforts and evaluations in general, Huffman Hayes *et al.* proposed an experimental framework for requirements tracing [21]. It is known that benchmarks can advance tool and technology development, which for instance has been the case for the Text REtrieval Conference (TREC), driving large-scale evaluations of IR methodologies [42]. On the other hand, benchmarking introduces a risk of over-engineering IR techniques on specific datasets. TREC enables generalizability by providing very large amounts of texts, but is still limited to certain domains such as news articles. Whether it is possible for the research community on traceability in software engineering to collect and distribute a similarly large dataset is an open question. Runeson *et al.* have discussed the meaning of benchmarks, however in relation to software testing [39]. They stress that researchers should first consider what the goal of the benchmarking is, and that a benchmark is a selected case that should be analyzed

[1] COEST, http://www.coest.org/

in its specific context. Consequently, as a benchmark is not an "average situation", they advocate that benchmarks should be studied with a qualitative focus rather than with the intention to reach statistical generalizability.

To enable categorization of evaluations of IR-based traceability recovery, including benchmarking studies, we adapted Ingwersen and Järvelins framework of IR evaluation contexts [24] into a taxonomy of IR-based traceability recovery evaluations. Within this context taxonomy, described in detail in Section 2, typical benchmarking studies belong to the innermost context. In the opposite end of the context taxonomy, the outermost evaluation context, industrial case studies reside. Furthermore, we add a dimension of study environment to the context taxonomy, to better reflect the external validity of the studied project, and its software artifacts.

To illustrate the context taxonomy with an example, and to evaluate the experimental framework proposed by Huffman Hayes *et al.* [21], we have conducted a quasi-experiment using the tools RETRO [23] and ReqSimile [35], using two sets of proprietary software artifacts as input. This constitutes a replication of earlier evaluations of RETRO [14, 44], as well as an extension considering both tools and input data. Also, we describe how it would be possible to extend this study beyond the innermost level of the context taxonomy, i.e., moving beyond technology-oriented evaluation contexts. Finally, we discuss concrete ways to strengthen the empirical evidence of IR-based traceability tools in relation to findings and suggestions by Falessi *et al.* [16], Runeson *et al.* [39], and Borg *et al.* [5, 6].

The paper is organized as follows. Section 2 gives a short overview of related work on IR-based traceability recovery in software engineering, while Section 3 describes our context taxonomy. Section 4 presents the research methodology, and the frameworks our experimentation follows. Section 4 presents the results from the experiments. In Section 6 we discuss the implications of our results, the validity threats of our study, and evaluations of IR-based traceability recovery in general. Finally, Section 7 presents conclusions and suggests directions of future research.

2 Background and Related work

During the last decade, several researchers proposed expressing traceability recovery as an IR problem. Proposed tools aim to support software engineers by presenting candidate traceability links ranked by textual similarities. The query is typically the textual content of a software artifact you want to link to other artifacts. Items in the search result can be either relevant or irrelevant, i.e., correct or incorrect traceability links are suggested.

In 2000, Antoniol *et al.* did pioneering work on traceability recovery when they used the standard *Vector Space Model* (VSM) [40] and the *binary independence model* [37] to suggest links between source code and documentation in

natural language [1]. Marcus and Maletic introduced *Latent Semantic Indexing* (LSI) [11] to recover traceability in 2003 [31]. Common to those papers is that they have a technical focus and do not go beyond reporting precision-recall graphs.

Other studies reported evaluations of traceability recovery tools using humans. Huffman Hayes *et al.* developed a traceability recovery tool named RETRO and evaluated it using 30 student subjects [23]. The students were divided into two groups, one working with RETRO and the other working manually. Students working with the tool finished a requirements tracing task faster and with a higher recall than the manual group, the precision however was lower. Natt och Dag *et al.* developed the IR-based tool ReqSimile to support market-driven requirements engineering and evaluated it in a controlled experiment with 23 student subjects [35]. They reported that subjects supported by ReqSimile completed traceability related tasks faster than subjects working without any tool support. De Lucia *et al.* conducted a controlled experiment with 32 students on the usefulness of tool-supported traceability recovery [9] and also observed 150 students in 17 software development projects [10]. In line with previous findings, they found that subjects using their tool completed a task, related to tracing various software artifacts, faster and more accurately than subjects working manually. They concluded that letting students use IR-based tool support is helpful when maintenance of traceability information is a process requirement.

Several previous publications have contributed to advancing the research on IR-based traceability recovery, either by providing methodological advice, or by mapping previous research. Huffman Hayes and Dekhtyar published a framework intended to advance reporting and conducting of empirical experiments on traceability recovery [21]. However, the framework has unfortunately not been applied frequently in previous evaluations, and the quality of reporting varies [6]. Another publication that offers structure to IR-based traceability recovery, also by Huffman Hayes *et al.*, distinguishes between *studies of methods* (are the tools capable of providing accurate results fast?) and *studies of human analysts* (how do humans use the tool output?) [22]. These categories are in line with experimental guidelines by Wohlin *et al.* [48], where the types of experiments in software engineering are referred to as either technology-oriented or human-oriented. Moreover, Huffman Hayes *et al.* propose assessing the accuracy of tool output, wrt. precision and recall, according to quality intervals named *Acceptable*, *Good*, and *Excellent*, based on the first author's practical experience of requirements tracing. Also discussing evaluation methodology, a recent publication by Falessi *et al.* proposes seven empirical principles for evaluating the performance of IR techniques [16]. Their work covers study design, statistical guidelines, and interpretation of results. Also, they present implementation strategies for the seven principles, and exemplify them in a study on industrial software artifacts originating from an Italian company.

Going beyond the simplistic measures of precision and recall is necessary to evolve IR tools [27], thus measures such as *Mean Average Precision* (MAP) [30],

Figure 1: The Integrated Cognitive Research Framework by Ingwersen and Järvelin [24], a framework for IR evaluations in context.

and *Discounted Cumulative Gain* (DCG) [26] have been proposed. To address this matter in the specific domain of IR-based traceability, a number of so called *secondary measures* have been proposed. Sundaram *et al.* developed *DiffAR*, *DiffMR*, *Lag*, and *Selectivity* [45] to assess the quality of generated candidate links.

In the general field of IR research, Ingwersen and Järvelin argue that IR is always evaluated in a context [24]. Their work extends the standard methodology of IR evaluation, the Laboratory Model of IR Evaluation developed in the Cranfield tests in the 60s [8], challenged for its unrealistic lack of user involvement [27]. Ingwersen and Järvelin proposed a framework, *The Integrated Cognitive Research Framework*, consisting of four integrated evaluation contexts, as presented in Figure 1. The innermost *IR context*, referred to by Ingwersen and Järvelin as "the cave of IR evaluation", is the most frequently studied level, but also constitutes the most simplified context. The *seeking context*, "drifting outside the cave", is used to study how users find relevant information among the information that is actually retrieved. The third context, the *work task context* introduces evaluations where the information seeking is part of a bigger work task. Finally, in the outermost realm, the *socio-organizational & cultural context*, Ingwersen and Järvelin argue that socio-cognitive factors are introduced, that should be studied in natural field experiments or studies, i.e., in-vivo evaluations are required. Moreover, they propose measures for the different evaluation contexts [24]. However, as those measures and the evaluation framework in itself are general and not tailored for neither software engineering nor traceability recovery, we present an adaptation in Section 3.

3 Derivation of Context Taxonomy

Based on Ingwersen and Järvelin's framework [24], we introduce a four-level context taxonomy in which evaluations of IR-based traceability recovery can be conducted, see Table 1. Also, we extend it by a dimension of evaluation environments motivated by our previous study [6], i.e., *proprietary*, *open source*, or *university*, as depicted in Figure 2. The figure also shows how the empirical evaluations pre-

sented in Section 2 map to the taxonomy. Note that several previous empirical evaluations of IR-based traceability recovery have used software artifacts from the open source domain, however, none of them were mentioned in Section 2.

We refer to the four integrated contexts as the *retrieval context* (Level 1), the *seeking context* (Level 2), the *work task context* (Level 3), and the *project context* (Level 4). Typical benchmarking experiments [1, 16, 31], similar to what is conducted within TREC, reside in the innermost retrieval context. Accuracy of tool output is measured by the standard IR-measures precision, recall, and F-score (defined in Section 4.2. In order to enable benchmarks in the seeking context, user studies or secondary measures are required [22, 45]. In both the two innermost contexts, traceability recovery evaluations are dominated by quantitative analysis. On the other hand, to study the findability offered by IR tools in the seeking context, defined as "the degree to which a system or environment supports navigation and retrieval" [34], researchers must introduce human subjects in the evaluations.

Regarding evaluations in the work task context, human subjects are necessary. Typically, IR-based traceability recovery in this context has been evaluated using controlled experiments with student subjects [9, 23, 35]. To assess the usefulness of tool support in work tasks involving traceability recovery, realistic tasks such as requirements tracing, change impact analysis, and test case selection should be studied in a controlled, in-vitro environment. Finally, in the outermost project context, the effect of deploying IR-based traceability recovery tools should be studied in-vivo in software development projects. Due to the typically low level of control in such study environments, a suitable evaluation methodology is a case study. An alternative to industrial in-vivo evaluations is to study student development projects, as De Lucia *et al.* [10] have done. In the work task context both quantitative and qualitative studies are possible, but in the project context qualitative analysis dominates. As applied researchers we value technology-oriented evaluations in the retrieval and seeking contexts, however, our end goal is to study IR tools in the full complexity of an industrial environment.

4 Method

We base discussions on the context taxonomy in Section 6 on a concrete study of traceability recovery. This section describes the definition, design and settings of the experimentation, organized into the four phases *definition*, *planning*, *realization* and *interpretation* as specified in the experimental framework by Huffman Hayes and Dekhtyar [21]. Also, our work followed the general experimental guidelines by Wohlin *et al.* [48]. According to the proposed context taxonomy in Figure 2, our experiment is an evaluation conducted in the retrieval context, i.e., in the cave, using datasets from two industrial contexts.

Evaluation Context	Description	Evaluation methodology	Example measures
Level 4: Project context	Evaluations in a socio-organizational context. The IR tool is studied when used by engineers within the full complexity of an in-vivo setting.	Case studies	Project metrics, tool usage
Level 3: Work task context	Humans complete real work tasks, but in an in-vitro setting. Goal of evaluation is to assess the casual effect of an IR tool when completing a task.	Controlled experiments, case studies	Work task results, time spent
Level 2: Seeking context	A seeking context with a focus on how the human finds relevant information among what was retrieved by the system.	Technology-oriented experiments	Usability, MAP, DCG, DiffAR, Lag
Level 1: Retrieval context	A strict retrieval context, performance is evaluated wrt. the accuracy of a set of search results.	Benchmarks	Precision, recall, F-measure

Table 1: Four integrated levels of context in IR-based traceability recovery evaluations.

4.1 Phase I: Definition

The definition phase presents the scope of the experimentation and describes the context. We entitle the study a *quasi-experiment* since there is no randomization in the selection of data sets nor IR tools.

Experiment Definition

The *goal* of the quasi-experiment is to evaluate traceability recovery tools in the context of embedded development, with the *purpose* of reporting results using proprietary software artifacts. The *quality focus* is to evaluate precision and recall of tools from the *perspective* of a researcher who wants to evaluate how existing approaches to traceability recovery perform in a specific industrial context.

Industrial Context

The software artifacts in the industrial dataset are collected from a large multinational company active in the power and automation sector. The context of the specific development organization within the company is safety critical embedded

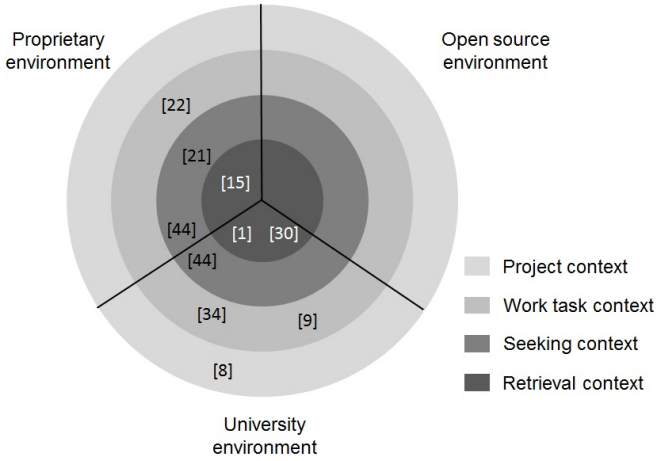


Figure 2: Contexts and environments in evaluations of IR-based traceability recovery. The numbers refer to references. Note that the publication by Sundaram *et al.* [45] contains both an evaluation in an industrial environment, as well as an evaluation in the university environment.

development in the domain of industrial control systems. The number of developers is in the magnitude of hundreds; a project has typically a length of 12-18 months and follows an iterative stage-gate project management model. The software is certified to a Safety Integrity Level (SIL) of 2 as defined by IEC 61508 [25], corresponding to a risk reduction factor of 1,000,000-10,000,000 for continuous operation. There are process requirements on maintenance of traceability information, especially between requirements and test cases. The software developers regularly perform tasks requiring traceability information, for instance when performing change impact analysis. Requirements and tests are predominantly specified in English NL text.

Characterization of Datasets

The first dataset used in our experiment originates from a project in the NASA Metrics Data Program, publicly available at COEST as the CM-1 dataset. The dataset specifies parts of a data processing unit and consists of 235 high-level requirements and 220 corresponding low-level requirements specifying detailed design. 361 traceability links between the requirement abstraction levels have been manually verified, out of 51 700 possible links. Items in the dataset have links to zero, one or many other items. This dataset is a de-facto benchmark of IR-based traceability recovery [6], thus we conduct a replication of previous evaluations.

Number of traceability links: 225		
Characteristic	Requirements	Test Case Descriptions
Items	224	218
Words	4 813	6 961
Words/Item	21.5	31.9
Avg. word length	6.5	7.0
Unique words	817	850
Gunning Fog Index	10.7	14.2
Flesch Reading Ease	33.7	14.8

Table 2: Descriptive statistics of the industrial data

Number of traceability links: 361		
Characteristic	High-level Reqs.	Low-level Reqs.
Items	235	220
Words	5 343	17 448
Words/Items	22.7	79.3
Avg. word length	5.2	5.1
Unique words	1 056	2 314
Gunning Fog Index	7.5	10.9
Flesch Reading Ease	67.3	59.6

Table 3: Descriptive statistics of the NASA data

The second dataset, referred to as the industrial data, consists of 225 requirements describing detailed design. These requirements are verified by 220 corresponding test case descriptions. The golden standard of 225 links was provided by the company, containing one link per requirement to a specific test case description. Thus, the link structure is different to the NASA data. The total number of combinatorial links is 49 500.

Both the NASA and the industrial datasets are bipartite, i.e., there exist only links between two subsets of software artifacts. Descriptive statistics of the datasets, calculated using the *Advanced Text Analyzer* at UsingEnglish.com[2], are presented in Tables 2 and 3. Calculating *Gunning Fog Index* [19] as a complexity metric for requirement specifications written in English has been proposed by Farbey [17]. The second complexity metric reported in Tables 2 and 3 is the *Flesch Reading Ease*. Wilson *et al.* have previously calculated and reported it for requirement specifications from NASA [47]. Both datasets are considered large according to the framework of Huffman Hayes and Dekhtyar [21], even though we would prefer to label them very small.

[2]http://www.usingenglish.com/members/text-analysis/

4.2 Phase II: Planning

This section describes the traceability recovery tools and the experimental design.

Description of Tools

We selected two IR-based traceability recovery tools for our experiment. Requirements Tracing on Target (RETRO) was downloaded from the library of Open Source Software from NASA's Goddard Space Flight Center[3], however only binaries were available. Source code and binaries of ReqSimile was downloaded from the source code repository SourceForge[4].

RETRO, developed by Huffman Hayes *et al.*, is a tool that supports software development by tracing textual software engineering artifacts [23]. The tool we used implements VSM with features having term frequency-inverse document frequency weights. Similarities are calculated as the cosine of the angle between feature vectors [3]. RETRO also implements a probabilistic retrieval model. Furthermore, the tool supports relevance feedback from users using the Standard Rochio feedback. Stemming is done as a preprocessing step and stop word removal is optional according to the settings in the tool. Later versions of RETRO also implement LSI, but were not available for our experiments. We used RETRO version V.BETA, Release Date February 23, 2006.

ReqSimile, developed by Natt och Dag *et al.*, is a tool with the primary purpose to provide semi-automatic support to requirements management activities that rely on finding semantically similar artifacts [35]. Examples of such activities are traceability recovery and duplicate detection. The tool was intended to support the dynamic nature of market-driven requirements engineering. ReqSimile also implements VSM and cosine similarities. An important difference to RETRO is the feature weighting; terms are weighted as $1 + log_2(freq)$ and no inverse document frequencies are considered. Stop word removal and stemming is done as preprocessing steps. In our experimentation, we used version 1.2 of ReqSimile.

To get yet another benchmark for comparisons, a tool was implemented using a naïve tracing approach as suggested by Menzies *et al.* [32]. The *Naïve* tool calculates the number of terms shared by different artifacts, and produces ranked lists of candidate links accordingly. This process was done without any stop word removal or stemming.

Experimental Design

The quasi-experiment has two independent variables: the IR-based traceability recovery tool used and the input dataset. The first one has three factors: RETRO, ReqSimile and Naïve as explained in Section 4.2. The second independent variable

[3] http://opensource.gsfc.nasa.gov/projects/RETRO/
[4] http://reqsimile.sourceforge.net/

has two factors: Industrial and NASA as described in Section 4.1. Consequently, six test runs were required to get a full factorial design.

IR-based traceability recovery tools are in the innermost evaluation context evaluated by verifying how many suggested links above a certain similarity threshold are correct, compared to a hand-crafted gold standard of correct links. Then, as presented in Figure 2, the laboratory model of IR evaluation is applied, thus recall and precision constitute our dependent variables. Recall and precision measure both the percentage of correct links recovered by a tool, and the amount of false positives. The aggregate measure F-score was also calculated, defined as the harmonic mean of precision and recall [3]:

$$F = 2 * \frac{precision * recall}{precision + recall}$$

Our null hypotheses, guiding the data analysis, are stated below. Performance is considered wrt. recall and precision.

NH1 The two tools implementing the vector space model, RETRO and ReqSimile, show equivalent performance.

NH2 Performance differences between the tools show equivalent patterns on the NASA and Industrial datasets.

NH3 RETRO and ReqSimile do not perform better than the Naïve tool.

4.3 Phase III: Realization

This section describes how the data was converted to valid input formats, and the actual tool usage.

Preprocessing of Datasets

Preprocessing the datasets was required since RETRO and ReqSimile use different input formats. The NASA dataset was available in a clean textual format and could easily be converted. The industrial data was collected as structured Microsoft Word documents including references, diagrams, revision history etc. We manually extracted the textual content and removed all formatting.

Experimental Procedure

Conducting the experiment consisted of six test runs combining all tools and datasets. In RETRO, two separate projects were created and the tool was run using default settings. We did not have access to neither a domain specific thesaurus nor a list of stop words. Relevance feedback using the standard Rochio method was not used. The *Trace All* command was given to calculate all candidate links, no filtering was done in the tool.

ReqSimile does not offer any configuration of the underlying IR models. The two input datasets were separated in two projects. After configuring the drivers of the database connections, the commands *Fetch requirements* and *Preprocess requirements* were given and the lists of candidate links were presented in the *Candidate requirements* tab.

The Naïve tool uses the same input format as RETRO. The tool does not have a graphical user interface, and was executed from a command-line interface.

5 Results and Interpretation

This section presents the results from our six test runs.

5.1 Phase IV: Interpretation

Huffman Hayes and Dekhtyar define the interpretation context as "the environment/circumstances that must be considered when interpreting the results of an experiment" [21]. We conduct our evaluation in the retrieval context as described in Section 3. Due to the small number of datasets studied, our hypotheses are not studied in a strict statistical context.

The precision-recall graphs and the plotted F-scores are used as the basis for our comparisons. All hypotheses do to some extent concern the concept of equivalence, which we study qualitatively in the resulting graphs. However, presenting more search results than a user would normally consider adds no value to a tool. We focus on the top ten search results, in line with recommendations from previous research [33,43,46], and common practise in web search engines. The stars in Figures 3 and 4 indicate candidate link lists of length 10.

The first null hypothesis stated that the two tools implementing the VSM show equivalent performance. Figures 3 and 5 show that RETRO and ReqSimile produce candidate links of equivalent quality, the stars are even partly overlapping. However, Figures 4 and 6 show that RETRO outperforms ReqSimile on the NASA dataset. As a result, the first hypothesis is rejected; *the two IR-based traceability recovery tools RETRO and ReqSimile, both implementing VSM, do not perform equivalently*.

The second null hypothesis stated that performance differences between the tools show equivalent patterns on the both datasets. The first ten datapoints of the precision-recall graphs, representing search hits of candidate links with lengths from 1 to 10, show linear quality decreases for both datasets. Graphs for the industrial data starts with higher recall values for short candidate lists, but drops faster to precision values of 5% compared to the NASA data. The Naïve tool performs better on the industrial data than on the NASA data, and the recall values increase at a higher pace, passing 50% at candidate link lists of length 10. The second hypothesis is rejected; *the tools show different patterns on the industrial dataset and the NASA dataset*.

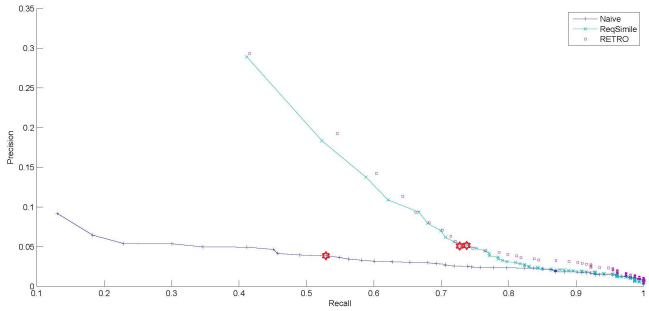


Figure 3: Precision-recall graph for the Industrial dataset. The stars show candidate link lists of length 10.

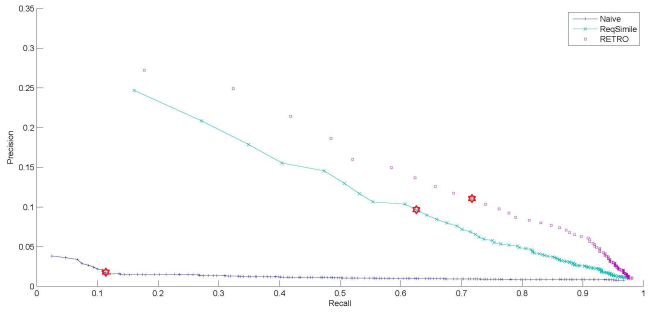


Figure 4: Precision-recall graph for the NASA dataset. The stars show candidate link lists of length 10.

The third null hypothesis, RETRO and ReqSimile do not perform better than the Naïve tool, is also rejected. Our results show that the Naïve tool, just comparing terms without any preprocessing, does not reach the recall and precision of the traceability recovery tools implementing VSM. *RETRO and ReqSimile perform better than the Naïve tool.*

6 Discussion

In this section, the results from the quasi-experiment and related threats to validity are discussed. Furthermore, we discuss how we could conduct evaluations in outer contextual levels based on this study, and we discuss how to advance evaluations of IR-based traceability recovery in general.

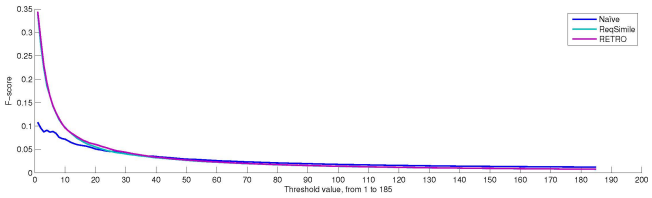


Figure 5: F-Score for the Industrial dataset. The X-axis shows the length of candidate link lists considered.

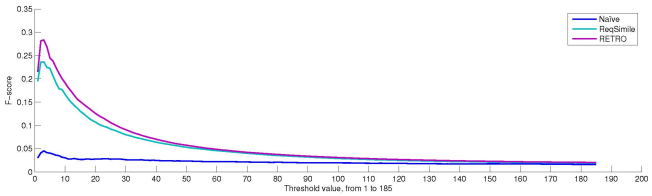


Figure 6: F-Score for the NASA dataset. The X-axis shows the length of candidate link lists considered.

6.1 Implication of Results

The IR-based traceability recovery tools RETRO and ReqSimile perform equivalently on the industrial dataset and similarly on the NASA data. From reading documentation and code of RETRO and ReqSimile, it was found that the tools construct different feature vectors. RETRO, but not ReqSimile, takes the inverse document frequency of terms into account when calculating feature weights. Consequently, terms overly frequent in the document set are not down-weighted as much in ReqSimile as in RETRO. This might be a major reason why RETRO generally performs better than ReqSimile in our quasi-experiment, even without the use of optional stop word removal. This shows that the construction of feature vectors is important to report when classifying traceability recovery tools, an aspect that often is omitted when reporting overviews of the field.

Our experimentation was conducted on two bipartite datasets of different nature. The NASA data has a higher density of traceability links and also a more complex link structure. RETRO and ReqSimile both perform better on the industrial dataset. The average amount of words of this dataset is fewer than in the NASA dataset, the reason for better IR performance is rather the less complex link structure. Not surprisingly, the performance of the traceability recovery is heavily dependant on the dataset used as input. Before there is a general large-scale dataset available for benchmarking, traceability recovery research would benefit from understanding various types of software artifacts. Especially for proprietary datasets used in experiments, characterization of both industrial context and the

dataset itself must be given proper attention.

As mentioned in Section 1, our quasi-experiment is partly a replication of studies conducted by Sundaram *et al.* [44], and Dekhtyar *et al.* [14]. Our results of using RETRO on the NASA dataset are similar, but not identical. Most likely, we have not applied the same version of the tool. Implementation of IR solutions forces developers to make numerous minor design decisions, i.e., details of the preprocessing steps, order of computations, numerical precision etc. Such minor variations can cause the differences in tool output we observe, thus version control of tools is important and should be reported.

6.2 Validity Threats

This section contains a discussion on validity threats to help define the creditability of the conclusions [48]. We focus on construct, internal and external validity.

Threats to *construct validity* concern the relationship between theory and observation. Tracing errors include both errors of inclusion and errors of exclusion. By measuring both recall and precision, the retrieval performance of a tool is well measured. However, the simplifications of the laboratory model of IR evaluation have been challenged [27]. There is a threat that recall and precision are not efficient measures of the overall usefulness of traceability tools. The question remains whether the performance differences, when put in a context with a user and a task, will have any practical significance. However, we have conducted a pilot study on RETRO and ReqSimile on a subset of the NASA dataset to explore this matter, and the results suggest that subjects supported by a slightly better tool also produce slightly better output [5].

Threats to *internal validity* can affect the independent variable without the researcher's knowledge and threat the conclusion about causal relationships between treatment and outcome. The first major threat comes from the manual preprocessing of data, which might introduce errors. Another threat is that the downloaded traceability recovery tools were incorrectly used. This threat was addressed by reading associated user documentation and running pilot runs on smaller dataset previously used in our department.

External validity concerns the ability to generalize from the findings. The bipartite datasets are not comparable to a full-size industrial documentation space and the scalability of the approach is not fully explored. However, a documentation space might be divided into smaller parts by filtering artifacts by system module, type, development team etc., thus also smaller datasets are interesting to study.

On the other hand, there is a risk that the industrial dataset we collected is a very special case, and that the impact of datasets on the performance of traceability recovery tools normally is much less. The specific dataset was selected in discussion with the company, to be representative and match our requirements on size and understandability. It could also be the case that the NASA dataset is not representative to compare RETRO and ReqSimile. The NASA data has been used

in controlled experiments of RETRO before, and the tool might be fine-tuned to this specific dataset. Consequently, RETRO and ReqSimile must be compared on more datasets to enable firm conclusions.

6.3 Advancing to outer levels

The evaluation we have conducted resides in the innermost retrieval context of the taxonomy described in Section 3. Thus, by following the experimental framework by Huffman Hayes *et al.* [21], and by using proprietary software artifacts as input, our contribution of empirical evidence can be classified as a Level 1 evaluation in an industrial environment, as presented in Figure 7. By adhering to the experimental framework, we provided enough level of detail in the reporting to enable future secondary studies to utilize our results.

Building upon our experiences from the quasi-experiment, we outline a possible research agenda to move the empirical evaluations in a more industry relevant direction. Based on our conducted Level 1 study, we could advance to outer levels of the context taxonomy. Primarily, we need to go beyond precision-recall graphs, i.e., step out of "the cave of IR evaluation". For example, we could introduce DCG as a secondary measure to analyze how the traceability recovery tools support finding relevant information among retrieved candidate links, repositioning our study as path A shows in the Figure 7.

However, our intention is to study how software engineers interact with the output from IR-based traceability recovery tools, in line with what we initially have explored in a pilot study [5]. Based on our experimental experiences, a future controlled experiment should be conducted with more subjects, and preferably not only students. An option would be to construct a realistic work task, using the industrial dataset as input, and run the experiment in a classroom setting. Such a research design could move a study as indicated by path B in Figure 7. Finally, to reach the outermost evaluation context as path C shows, we would need to study a real project with real engineers, or possibly settle for a student project. An option would be to study the information seeking involved in the state-of-practice change impact analysis process at the company from where the industrial dataset originates. The impact analysis work task involves traceability recovery, but currently the software engineers have to complete it without dedicated tool support.

6.4 Advancing traceability recovery evaluations in general

Our experiences from applying the experimental framework proposed by Huffman Hayes and Dekhtyar [21] are positive. The framework provided structure to the experiment design activity, and also it encouraged detailed reporting. As a result, it supports comparisons between experimental results, replications of re-

ported experiments, and it supports secondary studies to aggregate empirical evidence. However, as requirements tracing constitutes an IR problem (for a given artifact, relations to others are to be identified), it must be evaluated according to the context of the user as argued by Ingwersen and Järvelin [24]. The experimental framework includes "interpretation context", but it does not cover this aspect of IR evaluation. Consequently, we claim that our context taxonomy fills a purpose, as a complement to the more practical experimental guidelines offered by Huffman Hayes and Dekhtyar's framework [21].

While real-life proprietary artifacts are advantageous for the relevance of the research, the disadvantage is the lack of accessibility for validation and replication purposes. Open source artifacts offer in that sense a better option for advancing the research. However, there are two important aspects to consider. Firstly, open source development models tend to be different compared to proprietary development. For example, wikis and change request databases are more important than requirements documents or databases [41]. Secondly, there are large variations *within* open source software contexts, as there is within proprietary contexts. Hence, it is critical that research matches pairs of open source and proprietary software, as proposed by Robinson and Francis [38], based on several characteristics, and not only their being open source or proprietary. This also holds for generalization from studies from one domain to the other, as depicted in Figure 7.

Despite the context being critical, also evaluations in the innermost evaluation context can advance IR-based traceability recovery research, in line with the benchmarking discussions by Runeson *et al.* [39] and suggestions by members of the COEST [7, 12, 13]. Runeson *et al.* refer to the automotive industry, and argue that even though benchmarks of crash resistance are not representative to all types of accidents, there is no doubt that such tests have been a driving force in making cars safer. The same is true for the TREC conferences as mentioned in Section 1. Thus, the traceability community should focus on finding a series of meaningful benchmarks, including contextual information, rather than striving to collect a single large set of software artifacts to "rule them all". Regarding size however, such benchmarks should be considerably larger that the de-facto benchmarks used today. The same benchmark discussion is active within the research community on enterprise search, where it has been proposed to extract documents from companies that no longer exist, e.g., Enron [20], an option that might be possible also in software engineering.

Runeson *et al.* argue that a benchmark should not aim at statistical generalization, but a qualitative method of analytical generalization. Falessi *et al.* on the other hand, bring attention to the value of statistical hypothesis testing of tool output [16]. They reported a technology-oriented experiment in the seeking context (including secondary measures), and presented experimental guidelines in the form of seven empirical principles. However, the principles they proposed focus on the innermost contexts of the taxonomy in Figure 2, i.e., evaluations without human subjects. Also, since the independence between datapoints on a precision-

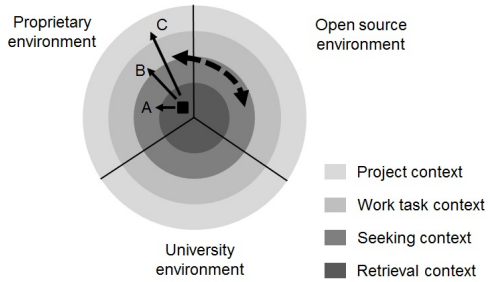


Figure 7: Our quasi-experiment, represented by a square, mapped to the taxonomy. Paths A-C show options to advance towards outer evaluation contexts, while the dashed arrow represents the possibility to generalize between environments as discussed by Robinson and Francis [38].

recall curve for a specific dataset is questionable, we argue that the result from each dataset instead should be treated as a single datapoint, rather than applying the cross-validation approach proposed by Falessi *et al.* As we see it, statistical analysis turns meaningful in the innermost evaluation contexts when we have access to sufficient numbers of independent datasets. On the other hand, when conducting studies on human subjects, stochastic variables are inevitably introduced, making statistical methods necessary tools.

Research on traceability recovery has the last decade, with a number of exceptions, focused more on tool improvements and less on sound empirical evaluations [6]. Since several studies suggest that further modifications of IR-based traceability recovery tools will only result in minor improvements [15, 36, 45], the vital next step is instead to assess the applicability of the IR approach in an industrial setting. The strongest empirical evidence on the usefulness of IR-based traceability recovery tools comes from a series of controlled experiments in the work task context, dominated by studies using student subjects [5, 9, 23, 35]. Consequently, to strengthen empirical evaluations of IR-based traceability recovery, we argue that contributions must be made along two fronts. Primarily, in-vivo evaluations should be conducted, i.e., industrial case studies in a project context. In-vivo studies on the general feasibility of the IR-based approach are conspicuously absent despite more than a decade of research. Thenceforth, meaningful benchmarks to advance evaluations in the two innermost evaluation contexts should be collected by the traceability community.

7 Conclusions and Future Work

We propose a context taxonomy for evaluations of IR-based traceability recovery, consisting of four integrated levels of evaluation contexts (retrieval, seeking, work

task, and project context), and an orthogonal dimension of study environments (university, open source, proprietary environment). To illustrate our taxonomy, we conducted an evaluation of the framework for requirements tracing experiments by Huffman Hayes and Dekhtyar [21].

Adhering to the framework, we conducted a quasi-experiment with two tools implementing VSM, RETRO and ReqSimile, on proprietary software artifacts from two embedded development projects. The results from the experiment show that the tools performed equivalently on the dataset with a low density of traceability links. However, on the dataset with a more complex link structure, RETRO outperformed ReqSimile. An important difference between the tools is that RETRO takes the inverse document frequency of terms into account when representing artifacts as feature vectors. We suggest that information about feature vectors should get more attention when classifying IR-based traceability recovery tools in the future, as well as version control of the tools. Furthermore, our research confirms that input software artifacts is an important factor in traceability experiments. Research on traceability recovery should focus on exploring different industrial contexts and characterize the data in detail, since replications of experiments on closed data are unlikely.

Following the experimental framework supported our study by providing structure and practical guidelines. However, it lacks a discussion on the evaluation contexts highlighted by our context taxonomy. On the other hand, when combined, the experimental framework and the context taxonomy offer a valuable platform, both for conducting and discussing, evaluations of IR-based traceability recovery.

As identified by other researchers, the widely used measures recall and precision are not enough to compare the results from tracing experiments [22]. The laboratory model of IR evaluation has been questioned for its lack of realism, based on progress in research on the concept of relevance and information seeking [27]. Critics claim that real human users of IR systems introduce non-binary, subjective and dynamic relevance, which affect the overall IR process. Our hope is that our proposed context taxonomy can be used to direct studies beyond "the cave" of IR evaluation, and motivate more industrial case studies in the future.

Acknowledgement

This work was funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering[5]. Special thanks go to the company providing the proprietary dataset.

[5] http://ease.cs.lth.se

Bibliography

[1] G. Antoniol, G. Canfora, G. Casazza, and A. De Lucia. Information retrieval models for recovering traceability links between code and documentation. In *Conference on Software Maintenance*, pages 40–49, 2000.

[2] G. Antoniol, G. Canfora, A. De Lucia, and E. Merlo. Recovering code to documentation links in OO systems. In *Proceedings of the 6th Working Conference on Reverse Engineering*, pages 136–144, 1999.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*. Addison-Wesley, 1999.

[4] E. Ben Charrada, D. Caspar, C. Jeanneret, and M. Glinz. Towards a benchmark for traceability. In *Proceedings of the 12th International Workshop on Principles on Software Evolution*, pages 21–30, 2011.

[5] M. Borg and D. Pfahl. Do better IR tools improve the accuracy of engineers' traceability recovery? In *Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering*, pages 27–34, 2011.

[6] M. Borg, K. Wnuk, and D. Pfahl. Industrial comparability of student artifacts in traceability recovery research - an exploratory survey. In *Proceedings of the 16th European Conference on Software Maintenance and Reengineering*, pages 181–190, 2012.

[7] J. Cleland-Huang, A. Czauderna, A. Dekhtyar, O. Gotel, J. Huffman Hayes, E. Keenan, J. Maletic, D. Poshyvanyk, Y. Shin, A. Zisman, G. Antoniol, B. Berenbach, A. Egyed, and P. Mäder. Grand challenges, benchmarks, and TraceLab: Developing infrastructure for the software traceability research community. In *Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*, 2011.

[8] C. Cleverdon. The significance of the Cranfield tests on index languages. In *Proceedings of the 14th Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1991.

[9] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Recovering traceability links in software artifact management systems using information retrieval methods. *Transactions on Software Engineering and Methodology*, 16(4), 2007.

[10] A. De Lucia, R. Oliveto, and G. Tortora. Assessing IR-based traceability recovery tools through controlled experiments. *Empirical Software Engineering*, 14(1):57–92, 2009.

[11] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[12] A. Dekhtyar and J. Huffman Hayes. Good benchmarks are hard to find: Toward the benchmark for information retrieval applications in software engineering. *Proceedings of the International Conference on Software Maintenance*, 2006.

[13] A. Dekhtyar, J. Huffman Hayes, and G. Antoniol. Benchmarks for traceability? In *Proceedings of the International Symposium on Grand Challenges in Traceability*, 2007.

[14] A. Dekhtyar, J. Huffman Hayes, and J. Larsen. Make the most of your time: How should the analyst work with automated traceability tools? In *Proceedings of the 3rd International Workshop on Predictor Models in Software Engineering*, 2007.

[15] D. Falessi, G. Cantone, and G. Canfora. A comprehensive characterization of NLP techniques for identifying equivalent requirements. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, 2010.

[16] D. Falessi, G. Cantone, and G. Canfora. Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. *Transactions on Software Engineering*, 2011.

[17] B. Farbey. Software quality metrics: considerations about requirements and requirement specifications. *Information and Software Technology*, 32(1):60–64, 1990.

[18] O. Gotel and C. Finkelstein. An analysis of the requirements traceability problem. In *Proceedings of the First International Conference on Requirements Engineering*, pages 94–101, 1994.

[19] R. Gunning. *Technique of clear writing - Revised edition*. McGraw-Hill, 1968.

[20] D. Hawking. Challenges in enterprise search. In *Proceedings of the 15th Australasian database conference*, pages 15–24, 2004.

[21] J. Huffman Hayes and A. Dekhtyar. A framework for comparing requirements tracing experiments. *Interational Journal of Software Engineering and Knowledge Engineering*, 15(5):751–781, 2005.

[22] J. Huffman Hayes, A. Dekhtyar, and S. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *Transactions on Software Engineering*, 32(1):4–19, 2006.

[23] J. Huffman Hayes, A. Dekhtyar, S. Sundaram, A. Holbrook, S. Vadlamudi, and A. April. REquirements TRacing on target (RETRO): improving software maintenance through traceability recovery. *Innovations in Systems and Software Engineering*, 3(3):193–202, 2007.

[24] P. Ingwersen and K. Järvelin. *The turn: Integration of information seeking and retrieval in context*. Springer, 2005.

[25] International Electrotechnical Commission. IEC 61508 ed 2.0, Electrical/-Electronic/Programmable electronic safety-related systems, 2010.

[26] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48, 2000.

[27] J. Kekäläinen and K. Järvelin. Evaluating information retrieval systems under the challenges of interaction and multidimensional dynamic relevance. *Proceedings of the COLIS 4 Conference*, pages 253—270, 2002.

[28] J. Lin, L. Chan, J. Cleland-Huang, R. Settimi, J. Amaya, G. Bedford, B. Berenbach, O. B Khadra, D. Chuan, and X. Zou. Poirot: A distributed tool supporting enterprise-wide automated traceability. In *Proceedings of the 14th International Conference on Requirements Engineering*, pages 363–364, 2006.

[29] M. Lormans, H-G. Gross, A. van Deursen, R. van Solingen, and A. Stehouwer. Monitoring requirements coverage using reconstructed views: An industrial case study. In *Procedings of the 13th Working Conference on Reverse Engineering*, pages 275–284, 2006.

[30] C. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[31] A. Marcus and J. Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. In *Proceedings of the International Conference on Software Engineering*, pages 125–135, 2003.

[32] T. Menzies, D. Owen, and J. Richardson. The strangest thing about software. *Computer*, 40(1):54–60, 2007.

[33] G. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.

[34] P. Morville. *Ambient findability: What we find changes who we become*. O'Reilly Media, 2005.

[35] J. Natt och Dag, T. Thelin, and B. Regnell. An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven product development. *Empirical Software Engineering*, 11(2):303–329, 2006.

[36] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *International Conference on Program Comprehension*, pages 68–71, 2010.

[37] S. E. Robertson and S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[38] B. Robinson and P. Francis. Improving industrial adoption of software engineering research: A comparison of open and closed source software. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, pages 21:1–21:10, 2010.

[39] P. Runeson, M. Skoglund, and E. Engström. Test benchmarks: What is the question? In *Proceedings of the International Conference on Software Testing Verification and Validation Workshop*, pages 368–371, 2008.

[40] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Commununications of the ACM*, 18(11):613–620, 1975.

[41] W. Scacchi. Understanding the requirements for developing open source software systems. *IEEE Software*, 149(1):24–39, 2002.

[42] A. Smeaton and D. Harman. The TREC experiments and their impact on europe. *Journal of Information Science*, 23(2):169–174, 1997.

[43] K. Spärck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: Development and comparative experiments. *Information Processing and Management*, 36(6):779–808, 2000.

[44] S. Sundaram, J. Huffman Hayes, and A. Dekhtyar. Baselines in requirements tracing. In *Proceedings of the Workshop on Predictor Models in Software Engineering*, pages 1–6, 2005.

[45] S. Sundaram, J. Huffman Hayes, A. Dekhtyar, and A. Holbrook. Assessing traceability of software engineering artifacts. *Requirements Engineering*, 15(3):313–335, 2010.

[46] T. Welsh, K. Murphy, T. Duffy, and D. Goodrum. Accessing elaborations on core information in a hypermedia environment. *Educational Technology Research and Development*, 41(2):19–34, 1993.

[47] W. Wilson, L. Rosenberg, and L. Hyatt. Automated analysis of requirement specifications. In *Proceedings of the 19th international conference on Software engineering*, pages 161–171, 1997.

[48] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: An introduction*. Kluwer Academic Publications, 1st edition, 1999.

PAPER IV

DO BETTER IR TOOLS IMPROVE THE ACCURACY OF ENGINEERS' TRACEABILITY RECOVERY?

Abstract

Large-scale software development generates an ever-growing amount of information. Multiple research groups have proposed using approaches from the domain of Information Retrieval (IR) to recover traceability. Several enhancement strategies have been initially explored using the laboratory model of IR evaluation for performance assessment. We conducted a pilot experiment using printed candidate lists from the tools RETRO and ReqSimile to investigate how different quality levels of tool output affect the tracing accuracy of engineers. Statistical testing of equivalence, commonly used in medicine, has been conducted to analyze the data. The low number of subjects in this pilot experiment resulted neither in statistically significant equivalence nor difference. While our results are not conclusive, there are indications that it is worthwhile to investigate further into the actual value of improving tool support for semi-automatic traceability recovery. For example, our pilot experiment showed that the effect size of using RETRO versus ReqSimile is of practical significance regarding precision and F-measure. The interpretation of the effect size regarding recall is less clear. The experiment needs to be replicated with more subjects and on varying tasks to draw firm conclusions.

Markus Borg, and Dietmar Pfahl, *In Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering*, 2011

1 Introduction

Development and maintenance of software often result in information overload. Knowledge workers are in general forced to spend more and more time to extract useful information. Maintaining traceability links between software artifacts is one approach to structure the information space of software development projects. Introducing information taxonomies and manually maintaining links is, however, an approach that does not scale very well. As a result, several researchers have proposed to support traceability recovery with tools based on IR methods, utilizing the fact that artifacts often have textual content in natural language.

Traceability recovery tools are generally evaluated by verifying how many suggested links above a certain similarity threshold are correct compared to a handcrafted set of correct links. The laboratory model of IR evaluation is applied to calculate the measures recall and precision, sometimes extended by a harmonic mean, the F-measure. Recall and precision are commonly reported using graphs, like the one shown in Figure 4. The "X" in Figure 4 marks the recall and precision values of RETRO [13] and ReqSimile [16] for the first ten links proposed by these tools. For "X", the recall and precision values of RETRO are 50% respectively 60% larger than those of ReqSimile. RETRO is a well-known traceability recovery tool. ReqSimile is a research tool developed at Lund University for the purpose to support information retrieval in the context of market-driven requirements engineering. However, since the real question is to what extent tools like RETRO and ReqSimile actually help engineers in performing traceability recovery tasks, one may wonder whether it is worthwhile to keep hunting for recall-precision improvements of traceability recovery tools.

To tackle this questions, we conducted a controlled experiment with 8 subjects to study how tool support affects the tracing process performed by engineers. The purpose of our experiment was to explore how the output from two traceability recovery tools, RETRO and ReqSimile, impacted a task requiring traceability information.

We analyzed our results using a test of equivalence, trying to prove that one treatment is indistinguishable from another. In equivalence testing, the null hypothesis is formulated such that the statistical test is a proof of similarity i.e., checking whether the tracing accuracies of engineers using different tools differ by more than a tolerably small amount Δ [20]. Our null hypothesis is the following:
Hypothesis: The engineers' accuracy of traceability recovery supported by RETRO *differs by more than* Δ compared to that supported by ReqSimile.

The alternative hypothesis is that the difference between the engineers' accuracies is smaller than Δ, which implies that the treatments can be considered equivalent. Accuracy is expressed in terms of recall and precision.

2 Related work

The last decade, several researchers proposed semi-automatic support to the task of traceability recovery. For example, traceability recovery tools have been developed implementing techniques based on algebraic or probabilistic models [1], data mining [23] and machine learning [19]. Several researchers have expressed the tracing task as an IR problem. The query in such a tool is typically the software artifact you want to link to other artifacts. The answer to a query is normally a ranked list of artifact suggestions, most often sorted by the level of textual similarity. The ranked list is analogous to the output of search engines used on the web. Items in the list can be either relevant or irrelevant for the given task.

In 2000, Antoniol *et al.* did pioneering work on traceability recovery when they used the standard vector space model (VSM) and probabilistic models to suggest links between source code and documentation in natural language [1]. Marcus and Maletic introduced Latent Semantic Indexing (LSI), another vector space approach, to recover traceability in 2003 [15]. Their work showed that LSI can achieve good results without the need for stemming, which is fundamental in VSM and the probabilistic models. The same year Spanoudakis *et al.* used a machine learning approach to establish traceability links [19]. By generating traceability rules from a set of artifacts given by the user, links were derived in the document set. Zhang *et al.* proposed automatic ontology population for traceability recovery [23]. They developed a text mining system to semantically analyze software documents. Concepts discovered by the system were used to populate a documentation ontology, which was then aligned with a source code ontology to establish traceability links.

Common to those papers is that they have a technical focus and present no or limited evaluations using software engineers solving real tasks. The majority of the published evaluations of traceability tools do not go beyond reporting recall-precision graphs or other measures calculated without human involvement. Exceptions include studies comparing subjects working with tool support to manual control groups. Huffman Hayes *et al.* developed a traceability recovery tool named RETRO and evaluated it using 30 student subjects [13]. The students were divided into two groups, one working with RETRO and the other working manually. Students working with the tool finished a requirements tracing task faster and with a higher recall than the manual group, the precision however was lower. De Lucia *et al.* conducted a controlled experiment with 32 students on the usefulness of supported traceability recovery [9]. They found that subjects using their tool completed a task related to tracing various software artifacts faster and more accurately than subjects working manually, i.e. without any support from a dedicated traceability recovery tool. In another study, De Lucia *et. al* observed 150 students in 17 software development projects and concluded that letting them use IR-based tool support is helpful when maintenance of traceability information is a process requirement [10]. An experiment similar to ours was conducted by Cud-

deback *et. al*, using students and student artifacts [8]. They had 26 subjects vet candidate requirements traceability matrices (RTMs) of varying accuracy. They concluded that subjects receiving the most inaccurate RTMs drastically improved them and that subjects in general balanced recall and precision.

Several researchers proposed ways to obtain better tool output, either by enhancing existing tools implementing standard IR techniques, or by exploring new or combined approaches. Using a thesaurus to deal with synonymy is one proposed enhancement strategy explored by different researchers [12, 18]. Zou *et al.* investigated term based improvement strategies such as including a part-of-speech tagger to extract key phrases and using a project glossary to weight certain terms higher [24]. Recently, Cleland-Huang *et al.* [6] and Asuncion *et al.* [2] used a machine learning approach, Latent Direchlet Allocation, to trace requirements. Furthermore, Chen has done preliminary work on combining IR-methods and text mining in a traceability recovery tool and reported improved results [5].

Even though enhancements lead to better tool outputs in certain cases, their general applicability and the benefit they generate for engineers performing a specific task remain uncertain. Oliveto *et al.* studied the impact of using four different methods for traceability recovery. In their empirical study, VSM, LSI and the Jensen-Shannon method resulted in almost equivalent results wrt. tracing accuracy [17]. LDA however, while not resulting in better accuracy, was able to capture different features than the others. As far as we know, no studies except Cuddeback *et. al* [8], have been published comparing how different quality levels of tool output impact of an engineer in a specific traceability task. If more empirical studies with humans were available, one could conduct a meta-analysis to investigate this matter. Since this is not the case, our approach is instead to compare in an experimental setting the effect of using support tools with differently accurate outputs on traceability tasks performed by humans.

3 Experimental Setup

This section describes the definition, design and setting of the experiment, following the general guidelines by Wohlin *et al.* [22]. An overview of our experimental setup is shown in Figure 1.

3.1 Experiment Definition and Context

The *goal* of the experiment was to study the tool-supported traceability recovery process of engineers, for the *purpose* of evaluating the impact of traceability recovery tools' accuracies, with respect to the engineers' accuracy of traceability recovery, from the *perspective* of a researcher evaluating whether quality variations between IR tool outputs significantly affect the tracing accuracy of engineers.

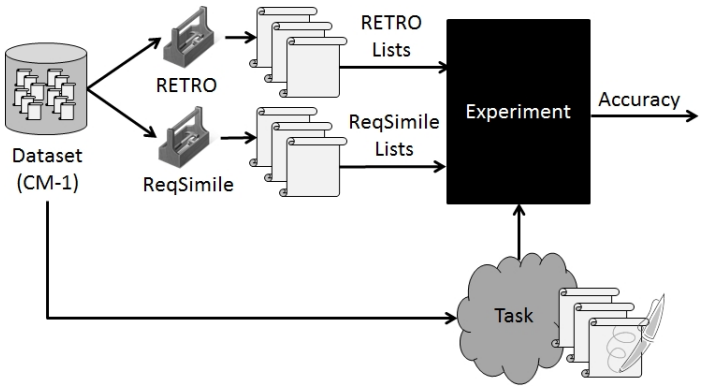


Figure 1: Overview of the experimental setup

3.2 Subjects and Experimental Setting

The experiment was executed at Lund University, Sweden. Eight subjects involved in software engineering research participated in the study. Six subjects were doctoral students, two subjects were senior researchers. Most subjects had industrial experience of software development.

The experiment was conducted in a classroom setting, the subjects worked individually. Each subject was randomly seated and supplied with a laptop with two electronic documents containing the artifacts that were to be traced in PDF format. Each subject also received a printed list per artifact to trace, containing candidate links as described in section 3.5. Four subjects received lists with candidate links generated by RETRO, the other four received candidate lists generated by ReqSimile. The lists were distributed randomly. The subjects received a pen, a two-page instruction, an answer sheet and a debriefing questionnaire. The subjects were supposed to navigate the PDF documents as they preferred, using the candidate link lists as support. All individual requirements were clickable as bookmarks, and keyword searching using the Find tool of their PDF viewer was encouraged.

3.3 Task and Description of the Dataset

It was decided to reuse a publicly available dataset and a task similar to previous tracing experiments to enable comparison to old results. The task, in which traceability recovery was required, was to estimate impact of a change request on the CM-1 dataset. For twelve given requirements, the subjects were asked to identify related requirements on a lower abstraction level. The task was given a realistic scenario involving time pressure, by having the subjects assume they should present their results in a meeting 45 minutes later. Before the actual experiment

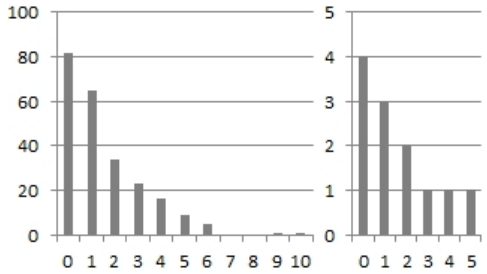


Figure 2: Histograms showing the link densities of CM-1 (left) and the subset used as the experimental sample (right).

started, the subjects were given a warm-up exercise to become familiar with the document structure and the candidate link lists.

The CM-1 data is a publicly available[1] set of requirements with complete traceability information. The data originates from a project in the NASA Metrics Data Program and has been used in several traceability experiments before [13, 14, 24]. The dataset specifies parts of a data processing unit and consists of 235 high-level requirements and 220 corresponding low-level requirements specifying detailed design. Many-to-many relations exist between abstraction levels. The link density of CM-1 and the representative subset used in the experiment are presented in Figure 2. This figure depicts histograms with the X-axis representing the number of low-level requirements related to one high-level requirement. Due to the rather unintuitive nature of the dataset, having many unlinked system requirements, the subjects received a hint saying that *"Changes to system requirements normally impact zero, one or two design items. Could be more, but more than five would really be exceptional"*.

Descriptive statistics of CM-1, including two commonly reported text complexity measures, are presented in Table 1. Farbey proposed calculating *Gunning Fog Index* as a complexity metric for requirement specifications written in English [11]. The second complexity metric reported is the *Flesch Reading Ease*, previously reported by Wilson *et al.* for requirement specifications from NASA [21].

3.4 Decription of the Tools

RETRO, developed by Huffman Hayes *et al.*, is a tool that supports software development by tracing textual software engineering artifacts [13]. The tool generates RTMs using standard information retrieval techniques. The evolution of RETRO accelerated when NASA analysts working on independent verification and validation projects showed interest in the tool. The version of the software we used implements VSM with features having term frequency-inverse document frequency

[1] www.coest.org

Number of traceability links: 361		
Characteristic	High-level Reqs.	Low-level Reqs.
Items	235	220
Words	5 343	17 448
Words/Items	22.7	79.3
Avg. word length	5.2	5.1
Unique words	1 056	2 314
Gunning Fog Index	7.5	10.9
Flesch Reading Ease	67.3	59.6

Table 1: Statistics of the CM-1 data, calculated using the *Text Content Analyzer* on UsingEnglish.com.

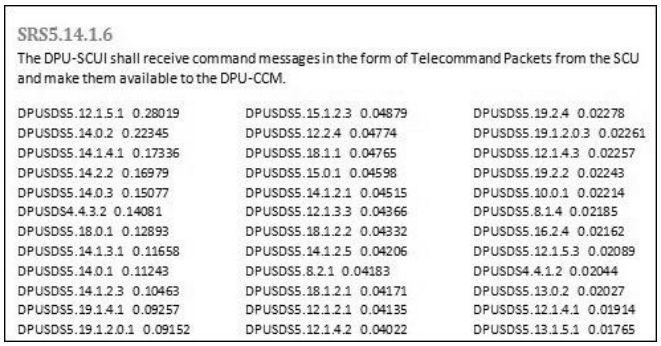
weights. Similarities are calculated as the cosine of the angle between feature vectors [3]. Stemming is done as a preprocessing step by default. For stop word removal, an external file must be provided, a feature we did not use. We used the RETRO version V.BETA, Release Date February 23, 2006.

ReqSimile, developed by Natt och Dag *et al.*, is a tool with the primary purpose to provide semi-automatic support to requirements management activities that rely on finding semantically similar artifacts [16]. Examples of such activities are traceability recovery and duplicate detection. The tool was intended to support the dynamic nature of market-driven requirements engineering. ReqSimile also implements VSM and cosine similarities. An important difference to RETRO is the feature weighting; terms are weighted as $1 + log_2(freq)$ and no inverse document frequencies are considered. Preprocessing steps in the tool include stop word removal and stemming. We used version 1.2 of ReqSimile.

3.5 Experimental Variables

In the context of the proposed experiment, the independent variable was the quality of the tool output given to the subjects. For each item to trace, i.e. for each high-level requirement, entire candidate link lists generated by the tools using default settings were used. No filtering was applied in the tools. The output varied between 47 and 170 items, i.e. each item representing a low-level requirement. An example of part of such a list is presented in Figure 3, showing high-level requirement SRS5.14.1.6 and the top part of a list of candidate low-level requirements and their cosine similarities. The two tools RETRO [13] and ReqSimile [16] are further described in Section 3.4. RETRO has outperformed ReqSimile wrt. accuracy of tool output in a previous experiment on the CM-1 dataset [4].

The lists were printed with identical formatting to ensure the same presentation. Thus, the independent variable was given two treatments, printed lists of candidate links ranked by RETRO (Treatment RETRO), and printed lists of candidate links ranked by ReqSimile (Treatment ReqSimile). The recall-precision graphs



SRS5.14.1.6

The DPU-SCUI shall receive command messages in the form of Telecommand Packets from the SCU and make them available to the DPU-CCM.

DPUSDS5.12.1.5.1 0.28019	DPUSDS5.15.1.2.3 0.04879	DPUSDS5.19.2.4 0.02278
DPUSDS5.14.0.2 0.22345	DPUSDS5.12.2.4 0.04774	DPUSDS5.19.1.2.0.3 0.02261
DPUSDS5.14.1.4.1 0.17336	DPUSDS5.18.1.1 0.04765	DPUSDS5.12.1.4.3 0.02257
DPUSDS5.14.2.2 0.16979	DPUSDS5.15.0.1 0.04598	DPUSDS5.19.2.2 0.02243
DPUSDS5.14.0.3 0.15077	DPUSDS5.14.1.2.1 0.04515	DPUSDS5.10.0.1 0.02214
DPUSDS4.4.3.2 0.14081	DPUSDS5.12.1.3.3 0.04366	DPUSDS5.8.1.4 0.02185
DPUSDS5.18.0.1 0.12893	DPUSDS5.18.1.2.2 0.04332	DPUSDS5.16.2.4 0.02162
DPUSDS5.14.1.3.1 0.11658	DPUSDS5.14.1.2.5 0.04206	DPUSDS5.12.1.5.3 0.02089
DPUSDS5.14.0.1 0.11243	DPUSDS5.8.2.1 0.04183	DPUSDS4.4.1.2 0.02044
DPUSDS5.14.1.2.3 0.10463	DPUSDS5.18.1.2.1 0.04171	DPUSDS5.13.0.2 0.02027
DPUSDS5.19.1.4.1 0.09257	DPUSDS5.12.1.2.1 0.04135	DPUSDS5.12.1.4.1 0.01914
DPUSDS5.19.1.2.0.1 0.09152	DPUSDS5.12.1.4.2 0.04022	DPUSDS5.13.1.5.1 0.01765

Figure 3: Example of top part of a candidate link list.

for the two tools on the experiment sample are presented in Figure 4, extended by the accuracy of the tracing results, i.e. the answer sets returned by subjects as described in Section 4.

The dependent variable, the outcome observed in the study, was the accuracy of the tracing result. Accuracy was measured in terms of recall, precision and F-measure. Recall measures the percentage of correct links traced by a subject, while precision measures the percentage of traced links that were actually correct. The F-measure is the harmonic mean of recall and precision. The time spent on the task was limited to 45 minutes, creating realistic time pressure. We also recorded the number of requirements traced by the subjects.

3.6 Experiment Design and Procedure

A completely randomized design was chosen. The experiment was conducted during one single session. The design was balanced, i.e. both treatments, RETRO and ReqSimile, were assigned to the same number of subjects. The two treatment were given to the subjects at random. Each subject received the same tasks and had not studied the system previously. When the 45 minutes had passed, the subjects were asked to answer a debriefing questionnaire.

3.7 Statistical Analysis

The null hypothesis was formulated as existence of a difference in the outcomes bigger than Δ. Δ defines the interval of equivalence, i.e., the interval where variation is considered to have no practical value. For this pilot study, we decided to set Δ to 0.05 for both recall, precision and F-measure. This means that finishing the task with 0.05 better or worse recall and precision does not have a practical value.

The two one-sided test (TOST) is the most basic form of equivalence testing used to compare two treatments. Confidence intervals for the difference between

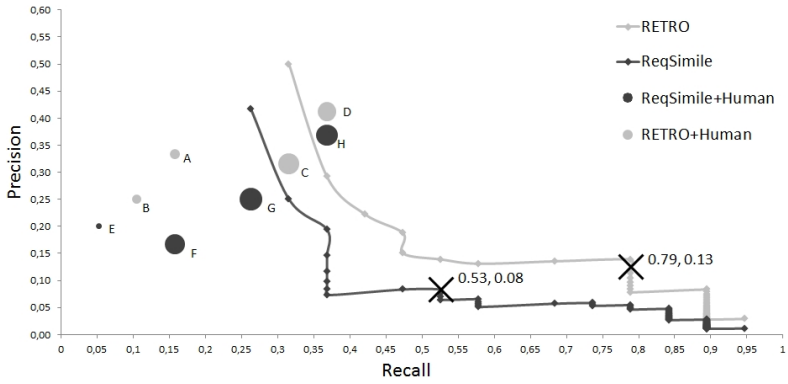


Figure 4: Recall-Precision graph for RETRO and ReqSimile for requirements tracing (our sample). The 'X'-symbols mark candidate link lists of length 10. Overall accuracy of answer sets returned by subjects is presented as circles, the diameter represents the relative number of links in the answer set. For a picture where also tool output is presented with relative sizes, see Figure 5.

two treatments must be defined. In a TOST analysis, a (1 - 2α)100% confidence interval is constructed [20]. We selected α = 0.05, thus we reject the null hypotheses that the outcomes of the treatments differ by at least Δ, if the 90% confidence interval for the difference is completely confined within the endpoints -Δ and +Δ. The 90% confidence intervals are calculated as follows:

$$point\_estimate\_outcome_{RETRO} - point\_estimate\_outcome_{ReqSimile} \pm 2.353\sqrt{std\_dev^2_{RETRO} + std\_dev^2_{ReqSimile}}$$

4 Results and Data Analysis

The experiment had no dropouts and as a result we collected 8 valid answer sheets and debriefing questionnaires. The answer sets were compared to the gold standard available for the datasets and the corresponding values for recall (Rc), precision (Pr) and F-measure (F) were calculated. The descriptive statistics for Rc, Pr, F, and the number of requirements traced are presented in Table 2. We also calculated the effect sizes using Cohen's d (cf. last column in Table 2). Results from the questionnaire are shown in Table 3.

Most subjects experienced the task as challenging and did not have enough time to finish. The list of common acronyms provided to assist the subjects, as was done in a previous case study using the CM-1 dataset [13], was not considered enough to appropriately understand the domain. Generally, the subjects considered

the printed candidate link lists as supportive and would prefer having tool support if performing a similar task in the future.

Table 4 characterizes the tool outputs of RETRO and ReqSimile as well as the tracing results provided by the subjects participating in the experiment. The upper part of the table shows the data for the treatment with RETRO, the lower part that for the treatment with ReqSimile. Each row in the table provides the following data: the ID of the high-level requirement (Req. ID), the number of low-level requirements suggested by the tool (#Links), the cosine similarities of the first and last link in the list of suggested low-level requirements (Sim. 1st link, Sim. last link), and for each subject (A to H) the number of reported links and the associated recall and precision (Sub. A: # / Rc / Pr). Bold values represent fully accurate answers. A hyphen indicates that a subject did not provide any data on that high-level requirement. IDs of high-level requirements printed in italics have no associated low-level requirements links, thus a correct answer would have been to report 0 links. For those requirements we define rc and pr equal to 1 if a subject actually reported 0 links, otherwise rc and pr equal 0. When subjects reported 0 links for high-level requirements that actually have low-level requirements, we define rc and pr equal to 0.

The number of high-level requirements the subjects had time to investigate during the experiment varied between three and twelve. On average, RETRO subjects investigated eight items and ReqSimile subjects investigated 8.75. All subjects apparently proceeded in the order the requirements were presented to them. Since subjects A and E investigated only three and four high-level requirements respectively, they clearly focused on quality rather than coverage. However, the precision of their tracing results does not reflect this focus. The mean recall for subjects supported by RETRO was higher than for subjects supported by ReqSimile, and also the mean precision. The standard deviations were however high, as expected when using few subjects. Not surprisingly, subjects reporting more links in their answer set reached higher recall values.

The debriefing questionnaire was also used to let subjects briefly describe their tracing strategies. Most subjects expressed focus on the top of the candidate lists. One subject reported the strategy of investigating the top 10 suggestions. Two subjects reported comparing similarity values and investigating candidate links until the first "big drop". Two subjects investigated links on the candidate lists until several in a row were clearly incorrect. Only one subject explicitly reported considering links after position 10. This subject investigated the first ten links, then every second until position 20, then every third until the 30th suggestion. This proved to be a a time-consuming approach and the resulting answer set was the smallest in the experiment. The strategies explained by the subjects are in line with our expectation that presenting more than 10 candidate links per requirement adds little value.

As Figure 4 shows, a naïve strategy of just picking the first one or two candidate links returned by the tools would in most cases result in better accuracy than

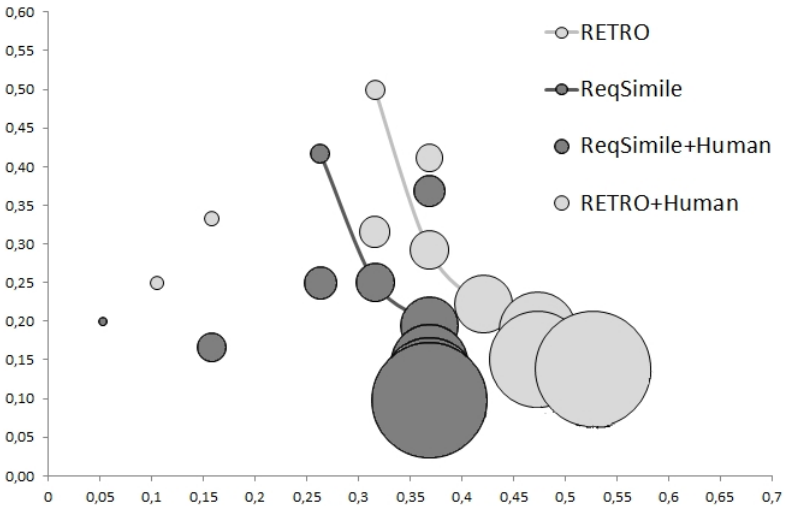


Figure 5: Circle diameters show relative number of links in answer sets. Tool output is plotted for candidate link lists of length from 1 to 6.

the subjects achieved. Also, there is a trend that subjects supported by RETRO handed in more accurate answer sets. Pairwise comparison of subjects ordered according to accuracy, i.e. B to E, A to F, C to G, D to H, indicates that the better accuracy of RETRO actually spills over to the subjects' tracing result.

Figure 5 shows relative sizes of answer sets returned by both human subjects and the tools, presenting how the number of tool suggestions grows linearly. The majority of human answer sets contained between one or two links per requirement, comparable to tools generating one or two candidate links.

The 90% confidence intervals of the differences between RETRO and ReqSimile are presented in Figure 6. Since none of the 90% confidence intervals of recall, precision, and F-measure are covered by the interval of equivalence, there is *no statistically significant equivalence of the engineers' accuracies of traceability recovery*, when using our choice of Δ. For completeness, we also did difference testing with the null hypothesis: The engineers' accuracy of traceability recovery supported by RETRO is equal to that supported by ReqSimile. This null hypothesis could not be rejected neither by a two-sided T-test nor a two-sided Wilcoxon rank-sum test with α=0.05. Consequently, there were *no statistically significant differences on the engineers' accuracies of traceability recovery* when supported by candidate link lists from different tools.

Our tests of significance are accompanied by effect-size statistics. Effect size is expressed as the difference between the means of the two samples divided by the root mean square of the variances of the two samples. On the basis of the effect size indices proposed by Cohen, effects greater or equal 0.5 are considered to be of

TREATMENT	**Reqs. Traced** (number)			
	Mean	Median	Std. Dev.	Eff. Size
RETRO	8.00	8.50	2.74	
ReqSimile	8.75	10.0	3.70	-0.230
	Recall			
	Mean	Median	Std. Dev.	Eff. Size
RETRO	0.237	0.237	0.109	
ReqSimile	0.210	0.211	0.118	0.232
	Precision			
	Mean	Median	Std. Dev.	Eff. Size
RETRO	0.328	0.325	0.058	
ReqSimile	0.247	0.225	0.077	1.20
	F-Measure			
	Mean	Median	Std. Dev.	Eff. Size
RETRO	0.267	0.265	0.092	
ReqSimile	0.218	0.209	0.116	0.494

Table 2: Descriptive statistics of experimental results.

medium size, while effect sizes greater or equal than 0.8 are considered large [7]. The effect sizes for precision and F-measure are high and medium respectively. Most researchers would consider them as being of practical significance. For recall, the effect size is too small to say anything conclusive.

5 Threats to Validity

The entire experiment was done during one session, lowering the risk of maturation. The total time for the experiment was less than one hour to help subjects keep focused. As the answers in the debriefing questionnaire suggests, it is likely that different subjects had different approaches to the process of artifact tracing, and the chosen approach might have influenced the outcome more than the different treatments. This is a threat to the internal validity. The fully randomized experiment design was one way to mitigate such effects. Future replications should aim at providing more explicit guidance to the subjects.

A possible threat to construct validity is that using printed support when tracing software artifacts is not representing how engineers would actually interact with the supporting IR tools, but it straightens the internal validity.

The CM-1 dataset used in the experiment, has been used in several previous tracing experiments and case studies. The dataset is not comparable to a large-scale industrial documentation space but is a representative subset. The CM-1 dataset originates from a NASA project, and is probably the most referenced dataset for requirements tracing. The subjects all do research in software engineering, most

QUESTIONS		
(1=Strongly agree, 5=Strongly disagree)	RETRO	ReqSimile
1. I had enough time to finish the task.	4.0	3.3
2. The list of acronyms gave me enough understanding of the domain to complete the task.	4.3	3.8
3. The objectives of the task were perfectly clear to me.	2.5	1.5
4. I experienced no major difficulties in performing the task.	3.3	4.3
5. The tool output (proposed links) really supported my task.	2.3	2.0
6. If I was performing a similar task in the future, I would want to use a software tool to assist.	2.3	1.8

Table 3: Results from the debriefing questionnaire. All questions were answered using a five-level Likert item. The means for each group are shown.

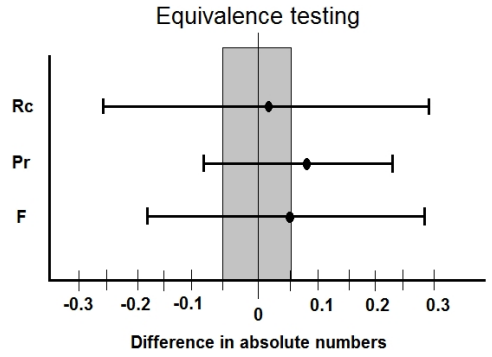


Figure 6: Differences in recall, precision and F-measure between RETRO and ReqSimile. The horizontal T-shaped bars depict confidence intervals. The interval of equivalence is the grey-shaded area.

Treatment RETRO							
Req. ID	**#Links**	**Sim. 1st link**	**Sim. last link**	**Sub. A # / Rc / Pr**	**Sub. B # / Rc / Pr**	**Sub. C # / Rc / Pr**	**Sub. D # / Rc / Pr**
SRS5.1.3.5	134	0.551	0.019	**1 / 1 / 1**	**1 / 1 / 1**	3 / 1 / 0.33	**1 / 1 / 1**
SRS5.1.3.9	116	0.180	0.005	4 / 0.2 / 0.25	1 / 0 / 0	1 / 0 / 0	2 / 0 / 0
SRS5.12.1.11	156	0.151	0.004	2 / 0 / 0	2 / 0 / 0	2 / 0 / 0	1 / 0 / 0
SRS5.12.1.8	125	0.254	0.005	2 / 0.5 / 0.5	0 / 0 / 0	3 / 0.5 / 0.33	0 / 0 / 0
SRS5.14.1.6	101	0.280	0.006	-	1 / 0 / 0	1 / 0.25 / 1	3 / 0.25 / 0.33
SRS5.14.1.8	117	0.173	0.005	-	1 / 0 / 0	**0 / 0 / 0**	1 / 0 / 0
SRS5.18.4.3	47	0.136	0.009	-	2 / 1 / 0.5	3 / 1 / 0.3	2 / 1 / 0.5
SRS5.19.1.10	135	0.140	0.004	-	-	1 / 0 / 0	1 / 0 / 0
SRS5.19.1.2.1	101	0.151	0.006	-	-	0 / 0 / 0	2 / 1 / 0.5
SRS5.2.1.3	127	0.329	0.005	-	-	3 / 0.67 / 0.67	4 / 0.67 / 0.5
SRS5.9.1.1	163	0.206	0.003	-	-	2 / 0 / 0	-
SRS5.9.1.9	159	0.240	0.005	-	-	-	-
Treatment ReqSimile							
Req. ID	**#Links**	**Sim. 1st link**	**Sim. last link**	**Sub. E # / Rc / Pr**	**Sub. F # / Rc / Pr**	**Sub. G # / Rc / Pr**	**Sub. H # / Rc / Pr**
SRS5.1.3.5	145	0.568	0.004	3 / 1 / 0.33	4 / 1 / 0.25	2 / 1 / 0.5	**1 / 1 / 1**
SRS5.1.3.9	142	0.318	0.029	1 / 0 / 0	2 / 0 / 0	3 / 0 / 0	1 / 0 / 0
SRS5.12.1.11	166	0.315	0.029	1 / 0 / 0	1 / 0 / 0	2 / 0 / 0	**0 / 1 / 1**
SRS5.12.1.8	111	0.335	0.022	-	0 / 0 / 0	1 / 0 / 0	4 / 0.5 / 0.25
SRS5.14.1.6	134	0.397	0.021	-	4 / 0.25 / 0.25	3 / 0.25 / 0.33	2 / 0.25 / 0.5
SRS5.14.1.8	170	0.397	0.029	-	2 / 0 / 0	2 / 0 / 0	2 / 0 / 0
SRS5.18.4.3	143	0.259	0.021	-	3 / 1 / 0.33	**1 / 1 / 1**	1 / 0 / 0
SRS5.19.1.10	160	0.340	0.025	-	2 / 0 / 0	**0 / 1 / 1**	3 / 0 / 0
SRS5.19.1.2.1	146	0.433	0.021	-	-	2 / 0 / 0	3 / 1 / 0.66
SRS5.2.1.3	151	0.619	0.018	-	-	2 / 0.66 / 1	1 / 0.33 / 1
SRS5.9.1.1	167	0.341	0.019	-	-	2 / 0 / 0	0 / 1 / 1
SRS5.9.1.9	157	0.527	0.018	-	-	1 / 0 / 0	**1 / 1 / 1**

Table 4: Characterization of tool outputs and tracing results provided by the subjects participating in the experiment.

have also worked as software developers in industry. Since the dataset is complex, dealing with embedded development in a safety critical domain, it was challenging for the subjects to fully understand the information they received to perform their task. In that regard, the subjects are comparable to newly employed software engineers in industry. This limits the generalizability of the experimental results, as software engineers normally would be more knowledgeable.

6 Discussion and Future Work

The results of the pilot experiment are inconclusive. The low number of subjects did not enable us to collect strong empirical evidence. The equivalence test (TOST) did not reject difference and the difference test (two-sided T-test) did not reject similarity. Thus, in this experiment, neither the evidence against equality nor difference was strong enough to reject either null hypothesis.

Although not statistically significant, we could see a trend that subjects supported with the better tool performed more accurately. Somewhat surprisingly, the precision of the subjects was not higher than that of the results the tools produced. For our specific task, with a time pressure, just using the tool output would generally be better than letting our subjects solve the task, using the tool output as support. One could argue that our subjects actually made the results worse. One direction of future research could be to explore under which circumstances this is the case.

Our experiment is in line with the finding of Cuddeback et. al [8], stating that subjects seem to balance recall and precision. We observed this trend in a very different experimental setup. Foremost, our task was to trace a subset of artifacts under time pressure using printed candidate link lists as support, as opposed to vet a complete RTM without time pressure using a tool. Other differences include: types of subjects and artifacts, and a sparser golden standard RTM.

Is it meaningful to study a tracing task with subjects that are not very knowledgeable in the domain? Embedded software development in the space industry is not easy to simulate in a classroom setting. Yet there is a need to understand the return on investment of improved accuracy of IR tools in support of traceability recovery. Controlled experiments can be one step forward. Our ambition is to replicate this pilot experiment using a larger number of student subjects, to explore whether any statistically significant results appear.

Recall and precision of traceability recovery tools are not irrelevant measures, but the main focus of research should be broader. For example, Figure 4 shows that the accuracy of RETRO is clearly better than that that of ReqSimile. However, the effect of using RETRO in a specific traceability recovery task is not that clear, as our pilot experiment suggests. Therefore, our future work aims at moving closer to research about information access, especially to the sub-domain of enterprise search, where more holistic approaches are explored. For example, we think that

assessing quality aspects such as findability in the software engineering context would mature the traceability recovery research.

Acknowledgements

This work was funded by the Industrial Excellence Center EASE - Embedded Applications Software Engineering[2]. Parts of this work were funded by the Excellence Center at Linköping-Lund in Information Technology (ELLIIT).

[2] http://ease.cs.lth.se

Bibliography

[1] G. Antoniol, G. Canfora, G. Casazza, and A. De Lucia. Information retrieval models for recovering traceability links between code and documentation. In *Conference on Software Maintenance*, pages 40–49, 2000.

[2] H. Asuncion, A. Asuncion, and R. Taylor. Software traceability with topic modeling. In *Proceedings of the International Conference on Software Engineering*, pages 95–104, 2010.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*. Addison-Wesley, 1999.

[4] L. Brodén. *Requirements traceability recovery: A study of available tools*. Master thesis, Lund University, http://sam.cs.lth.se/ExjobGetFile?id=377, 2011.

[5] X. Chen. Extraction and visualization of traceability relationships between documents and source code. In *Proceedings of the International Conference on Automated Software Engineering*, pages 505–509, 2010.

[6] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker. A machine learning approach for tracing regulatory codes to product specific requirements. In *Proceedings International Conference on Software Engineering*, pages 155–164, 2010.

[7] J. Cohen. *Statistical power analysis for the behavioral sciences*. Routledge, 1988.

[8] D. Cuddeback, A. Dekhtyar, and J. Huffman Hayes. Automated requirements traceability: The study of human analysts. In *Proceedings of the 18th International Requirements Engineering Conference*, pages 231–240, 2010.

[9] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Recovering traceability links in software artifact management systems using information retrieval methods. *Transactions on Software Engineering and Methodology*, 16(4), 2007.

[10] A. De Lucia, R. Oliveto, and G. Tortora. Assessing IR-based traceability recovery tools through controlled experiments. *Empirical Software Engineering*, 14(1):57–92, 2009.

[11] B. Farbey. Software quality metrics: considerations about requirements and requirement specifications. *Information and Software Technology*, 32(1):60–64, 1990.

[12] J. Huffman Hayes, A. Dekhtyar, and S. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *Transactions on Software Engineering*, 32(1):4–19, 2006.

[13] J. Huffman Hayes, A. Dekhtyar, S. Sundaram, A. Holbrook, S. Vadlamudi, and A. April. REquirements TRacing on target (RETRO): improving software maintenance through traceability recovery. *Innovations in Systems and Software Engineering*, 3(3):193–202, 2007.

[14] A. Mahmoud and N. Niu. Using semantics-enabled information retrieval in requirements tracing: An ongoing experimental investigation. In *Proceedings of the International Computer Software and Applications Conference*, pages 246–247, 2010.

[15] A. Marcus and J. Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. In *Proceedings of the International Conference on Software Engineering*, pages 125–135, 2003.

[16] J. Natt och Dag, V. Gervasi, S. Brinkkemper, and B. Regnell. A linguistic-engineering approach to large-scale requirements management. *IEEE Softw.*, 22(1):32–39, 2005.

[17] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *International Conference on Program Comprehension*, pages 68–71, 2010.

[18] R. Settimi, J. Cleland-Huang, O. Ben Khadra, J. Mody, W. Lukasik, and C. DePalma. Supporting software evolution through dynamically retrieving traces to UML artifacts. In *Proceedings of the 7th International Workhop on Principles of Software Evolution*, pages 49–54, 2004.

[19] G. Spanoudakis, A. d'Avila-Garcez, and A. Zisman. Revising rules to capture requirements traceability relations: A machine learning approach. In *Proceedings of the 15th International Conference in Software Engineering and Knowledge Engineering*, 2003.

[20] S. Wellek. *Testing statistical hypotheses of equivalence*. Chapman and Hall, 2003.

[21] W. Wilson, L. Rosenberg, and L. Hyatt. Automated analysis of requirement specifications. In *Proceedings of the 19th international conference on Software engineering*, pages 161–171, 1997.

[22] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: An introduction*. Kluwer Academic Publications, 1st edition, 1999.

[23] Y. Zhang, R. Witte, J. Rilling, and V. Haarslev. Ontological approach for the semantic recovery of traceability links between software artefacts. *IET Software*, 2(3):185–203, 2008.

[24] X. Zou, R. Settimi, and J. Cleland-Huang. Improving automated requirements trace retrieval: A study of term-based enhancement methods. *Empirical Software Engineering*, 15(2):119–146, 2010.