Empir Software Eng (2014) 19:1565–1616
DOI 10.1007/s10664-013-9255-y

Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability

Markus Borg · Per Runeson · Anders Ardö

Published online: 9 May 2013

Abstract 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. The objective of this study is to produce a map of work on IR-based trace recovery, with a particular focus on previous evaluations and strength of evidence. We conducted a systematic mapping of IR-based trace recovery. Of the 79 publications classified, a majority applied algebraic IR models. 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. 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.

Keywords Traceability · Information retrieval · Software artifacts · Systematic mapping study

Communicated by: Giulio Antoniol

M. Borg (✉) · P. Runeson
Department of Computer Science, Lund University, Lund, Sweden
e-mail: markus.borg@cs.lth.se

P. Runeson
e-mail: per.runeson@cs.lth.se

A. Ardö
Department of Electrical and Information Technology, Lund University, Lund, Sweden
e-mail: anders.ardo@eit.lth.se

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 (Olsson 2002; Cleland-Huang et al. 2003). 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 (Gotel and Finkelstein 1994; Huffman Hayes et al. 2006*).[1] With access to trace information, engineers can more efficiently perform work tasks such as impact analysis, identification of reusable artifacts, and requirements validation (Antoniol et al. 2002*; Winkler and Pilgrim 2010). Furthermore, research has identified lack of traceability to be a major contributing factor in project overruns and failures (Gotel and Finkelstein 1994; Dömges and Pohl 1998; Cleland-Huang et al. 2003). Moreover, as traceability plays a role in software verification, safety standards such as ISO 26262 (International Organization for Standardization 2011) for the automotive industry, and IEC 61511 (International Electrotechnical Commission 2003) for the process industry, mandate maintenance of traceability information (Katta and Stålhane 2011), as does the CMMI process improvement model (Carnegie Mellon Software Engineering Institute 2010). However, manually maintaining trace links is an approach that does not scale (Heindl and Biffl 2005). In addition, the dynamics of software development makes it tedious and error-prone (Dömges and Pohl 1998; Huffman Hayes et al. 2006*; Falessi et al. 2010).

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 (Antoniol et al. 2002*; Marcus and Maletic 2003; De Lucia et al. 2004*; Huffman Hayes et al. 2006*; Lormans and van Deursen 2006*). Also, other studies have reported that the use of semi-automated trace recovery reduces human effort when performing requirements tracing (Huffman Hayes et al. 2006*; Natt och Dag et al. 2006*; De Lucia et al. 2006b, 2007*, 2009*a). 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 (Kitchenham and Charters 2007). 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 (Gotel et al. 2012), organized by

[1] We use an asterisk ('*') to distinguish primary publications in the systematic mapping from general references.

the Center of Excellence for Software Traceability (CoEST).[2] Furthermore, it could assess the recent claims that applying more advanced IR models does not improve results (Oliveto et al. 2010*; Falessi et al. 2010).

We have conducted a Systematic Mapping (SM) study (Kitchenham et al. 2011; Petersen et al. 2008) that clusters publications on IR-based trace recovery. SMs and Systematic Literature Reviews (SLR) are primarily distinguished by their driving Research Questions (RQ) (Kitchenham et al. 2011), 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 SM in three RQs:

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?

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 SM was conducted. Section 5 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 (Cleland-Huang et al. 2012). 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)*" (Manning et al. 2008). 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

[2] www.coest.org.

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

Retrieval models			Misc.		
Algebraic models	Probabilistic models	Statistical language models	Weighting schemes	Similarity measures / distance functions	Enhancement strategies
Vector space model (VSM)	Binary independence model (BIM)	Language model (LM)	Binary	Cosine similarity	Relevance feedback
Latent semantic indexing (LSI)	Probabilistic inference network (PIN)	Probabilistic latent semantic indexing (PLSI)	Raw	Dice's coefficient	Thesaurus
	Best match 25 (BM25)[a]	Latent dirichlet allocation (LDA)	Term frequency inverse document frequency (TFIDF)	Jaccard index	Phrasing
		Correlated topics model (CTM)	Best match 25 (BM25)[a]	Jensen-Shannon divergence (JS)	Clustering
		Relational topics model (RTM)			

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

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 (2001): "*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. (2010) 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*" (Kruchten 2004), 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*" (Cleland-Huang et al. 2012). 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 (Baeza-Yates and Ribeiro-Neto 2011), 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 (Manning et al. 2008).

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* (BoW) model, a simplifying assumption that represents a document as an unordered collection of words, disregarding word order (Manning et al. 2008). 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 (Zhai 2007). The most well-known algebraic model is the commonly applied *Vector Space Model (VSM)* (Salton et al. 1975), 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 (Singhal 2001). 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 (Manning et al. 2008) have been applied. In an attempt to reduce the noise of NL (such as synonymy and polysemy), *Latent Semantic Indexing* (LSI) was introduced (Deerwester et al. 1990). 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 (Zhai 2007). In IR-based trace recovery, this is commonly implemented using the Standard Rocchio method (Rocchio 1971). 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 (Singhal 2001). Documents are then ranked according to their probability of being relevant (Maron and Kuhns 1960), referred to as the probabilistic ranking principle (Robertson 1977). In trace recovery, the *Binary Independence Retrieval* model (BIM) (Robertson and Jones 1976) was first applied to establish links. BIM naïvely assumes that terms are independently distributed, and essentially applies the Naïve Bayes classifier for document ranking (Lewis 1998). Different weighting schemes have been explored to improve results, and currently the *BM25* weighting used in the non-binary Okapi system (Robertson and Zaragoza 2009) 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) (Turtle and Croft 1991). In an inference network, relevance is modeled by the uncertainty associated with inferring the query from the document (Zhai 2007). 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 (Zhai 2007), 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 (Ponte and Croft 1998). A refinement of simple LMs, topic models, describes documents as a mixture over topics. Each individual topic is then characterized by an LM (Zhai 2008). In trace recovery research, studies applying the four topic models *Probabilistic Latent Semantic Indexing* (PLSI) (Hofman 2001), *Latent Dirichlet Allocation* (LDA) (Blei et al. 2003), *Correlated Topic Model* (CTM) (Blei and Lafferty 2007) and *Relational Topic Model* (RTM) (Chang and Blei 2010) 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* (Zhai and Lafferty 2001).

Several attempts are made to improve an IR model, in this paper referred to as *enhancement strategies*. Apart from the already described relevance feedback, one of the most common approaches in IR is to introduce a *thesaurus*. A thesaurus is a tool for vocabulary control, typically defined for a specific subject area, such as art or biology, formally organized so that a priori relationships between concepts are made explicit (Aitchison et al. 2000). Standard usage of a thesaurus is to provide an IR system with *preferred and non-preferred terms*, restricted vocabularies of terms that the IR system is allowed to, or not allowed to, use for indexing and searching, and *semantic relations*, relations between terms such as synonymy and hyponymy. Another enhancement strategy in IR is *phrasing*, an approach to exceed indexing according to the BoW model (Croft et al. 1991). A phrase is a sequence of two or more words, expected to be more accurate in representing document content than independent words. Detecting phrases for indexing can be done using either a statistical analysis of term frequency and co-occurrence, or by a syntactical approach, i.e. analyzing grammatical structure using a parts-of-speech tagger. Yet another enhancement strategy is *clustering*, based on the hypothesis that “documents relevant to a query tend to be more similar to each other than to irrelevant documents and hence are likely to be clustered together” (Charikar et al. 1997). Clustering can be used for different purposes, e.g. presenting additional search results or to structure the presentation of search results.

Finally, a number of measures used to evaluate IR tools have been defined. Accuracy of a set of search results is primarily measured by the standard IR-measures *precision* (the fraction of retrieved instance 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) (Baeza-Yates and Ribeiro-Neto 2011). 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) (Antoniol et al. 2002*) and *Selectivity* (Sundaram et al. 2010*). *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 (Manning et al. 2008), and *Discounted Cumulative Gain* (DCG) (Järvelin and Kekäläinen 2000) (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. (2010*) 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. (2012) describe this process as a sequence of four key steps, where the fourth step requires human judgement. 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
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. Concrete examples, put in work task contexts, are presented in Section 3.4.

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 (De Lucia et al. 2006*b, 2008*). 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 (2011*), several subjects described such an approach to deal with tool output, even without explicit instructions. Coverage analysis is another strategy proposed by De Lucia et al. (2009*b), intended to follow up on the step of iteratively decreasing the similarity threshold. 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. (1992), in which a tool suite analyzing NL requirements was developed. The tool suite parsed NL requirements to build semantic representations, and used artificial intelligence approaches to help engineers establish trace links between individual requirements (Bras and Toussaint 1993). Apart from analyzing relations between artifacts, the tools evaluated consistency, completeness, verifiability and modifiability (Castell et al. 1994). In 1998, a study by Fiutem and Antoniol (1998) presented a recovery process to bridge the gap between design and code, based on edit distances between artifacts. 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 (Antoniol et al. 2000). Their milestone work from 2002 compared two standard IR models, probabilistic retrieval using the BIM and the VSM (Antoniol et al. 2002*). Simultaneously, in the late 1990s, Park et al. (2000*) worked on tracing dependencies between artifacts using a sliding window combined with syntactic parsing. 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. (2002*) did research on requirement dependencies in the dynamic environment of market-driven requirements engineering. They developed the tool ReqSimile, implementing trace recovery based on the VSM, and later evaluated it in a controlled experiment (Natt och Dag et al. 2006*). A publication by Marcus and Maletic (2003), the second most cited article in the field, constitutes a technical milestone in IR-based trace recovery. 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 et al. (2004*) enhanced VSM retrieval with relevance feedback and introduced secondary performance metrics. From early on, their research had a human-oriented perspective, aimed at supporting V&V activities at NASA using their tool RETRO (Huffman Hayes et al. 2007*).

De Lucia et al. (2005*) have conducted work focused on empirically evaluating LSI-based trace recovery in their document management system ADAMS. They have advanced the empirical foundation by conducting a series of controlled experiments and case studies with student subjects (De Lucia et al. 2006*a, 2007*, 2009*a). 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 (Lin et al. 2006). Much of their work has focused on improving the accuracy of their tool by enhancements such as: applying a thesaurus to deal with synonymy (Settimi et al. 2004*), extraction of key phrases (Zou et al. 2010*), and using a project glossary to weight the most important terms higher (Zou et al. 2010*).

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. (2007*b) combined several IR models using a voting scheme, including the probabilistic topic model Latent Dirachlet Allocation (LDA). Parvathy et al. (2008*) proposed using the Correlated Topic Model (CTM), and Gethers et al. (2011*) suggested using Relational Topic Model (RTM). Abadi et al. (2008*) proposed using Probabilistic Latent Semantic Indexing (PLSI) and utilizing two concepts based on information theory, Sufficient Dimensionality Reduction (SDR) and Jensen-Shannon Divergence (JS). Capobianco et al. (2009*b) proposed representing NL artifacts as B-splines and calculating similarities as distances between them on the Cartesian plane. Sultanov and Huffman Hayes (2010*) implemented trace recovery using a swarm technique, 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

In the beginning of 2012, a textbook on software traceability edited by Cleland-Huang et al. (2012) was published. Presenting software traceability from several perspectives, the book contains a chapter authored by De Lucia et al. (2012) specifically dedicated to IR-based trace recovery. In the chapter, the authors thoroughly present an overview of the field including references to the most important work. Also, the chapter constitutes a good introduction for readers new to the approach, as it describes the basics of IR models. Consequently, the book chapter by De Lucia et al. is closely related to our work. However, our work has different characteristics. First, De Lucia et al.'s work has more the character of a textbook, including enough background material on IR, as well as examples of applications in context, to introduce readers to the field as a stand-alone piece of work. Our systematic mapping on the other hand, is not intended as an introduction to the field of IR-based trace recovery, but requires extensive pre-understanding. Second, while De Lucia et al. report a large set of references to previous work, the method used to identify previous publications is not reported. Our work instead follows the established guidelines for SMs (Kitchenham and Charters 2007), and reports from every phase of the study in a detailed protocol.

Furthermore, basically every publication on IR-based trace recovery contains some information on previous research in the field. Another good example of a summary of the field was provided by De Lucia et al. (2009*a). 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 SM. De Lucia et al. (2008) have also surveyed proposed approaches to traceability management for impact analysis. They discussed previous work based on a conceptual framework by Bianchi et al. (2000), 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 (2010) have presented a survey of IR-based trace recovery as part of an overview of applications of IR in software engineering. 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 (2006*) presented a taxonomy of IR models. 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. (2010) recently published a comprehensive taxonomy of IR techniques available to identify equivalent requirements. They adopted the term variation point from Software Product Line Engineering (Pohl et al. 2005), 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 mapping study 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).

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 (Marcus et al. 2004). 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 (Gay et al. 2009). 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 (2010). 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. (2011) recently published a literature review on feature location.

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 (2005a). 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 (Borg et al. 2012b). Huffman Hayes et al. (2006*) 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?). Furthermore, they proposed assessing the accuracy of tool 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.

Considering empirical evaluations, we extend the classifications proposed by Huffman Hayes et al. (2006*) by an adapted version of the *Integrated Cognitive Research Framework* by Ingwersen and Järvelin (2005). 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 (Cleverdon 1991), challenged for its unrealistic lack of user involvement (Kekäläinen and Järvelin 2002). 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 Fig. 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

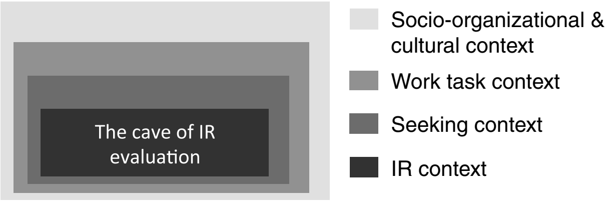

The cave of IR evaluation
Socio-organizational & cultural context
Work task context
Seeking context
IR context

Fig. 1 The integrated cognitive research framework by Ingwersen and Järvelin (2005), a framework for IR evaluations in context

presented in Fig. 12 in Section 5. For more information on the context taxonomy, we refer to our original publication (Borg et al. 2012a).

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 (Runeson et al. 2012).

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

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

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

> 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 (Wohlin et al. 2012).

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. (2012) present a literature review on characteristics of artifacts reported to impact trace recovery evaluations, 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. (2012b) recently highlighted that a majority of previous evaluations of IR-based trace recovery have been conducted using artifacts developed by students. 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 values from evaluations in the retrieval and seeking contexts. A number of publications, including the pioneering work by Antoniol et al. (2002*), used the traditional style from the ad hoc retrieval task organized by the Text REtrieval Conference (TREC) (Voorhees 2005), 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 Fig. 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

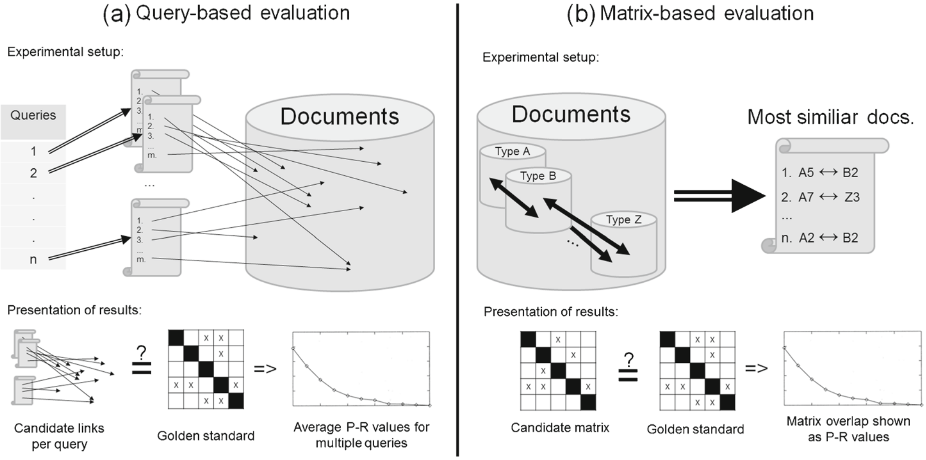


Fig. 2 Query-based evaluation vs. matrix-based evaluation of IR-based trace recovery

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 (Antoniol et al. 2002*; Li et al. 2008*; Borg and Pfahl 2011*).

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 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 a candidate traceability matrix. We refer to this reporting style as *matrix-based* evaluation. The candidate traceability matrix can be compared to a gold standard, and the accuracy (i.e. overlap between the matrices) can be presented as a P-R curve, as shown in (b) in Fig. 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. (2006*) defined the quality intervals described in Section 3.3 to support this evaluation style.

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. *Variable threshold*, a new similarity interval is defined by the minimum and maximum similarities (i.e. similarities are normalized against the highest and lowest similarities), and either a percentage of the candidate trace links are selected or a new constant threshold is introduced.

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 (Borg and Pfahl 2011*), 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.

The two inherently different use cases of an IR-based trace recovery tool, reflected by the split into matrix-based and query-based evaluations, also call for different evaluations regarding cut-off strategies. The first main use case of an IR-based trace recovery tool is when one or more candidate trace links from a specific artifact are requested by an engineer. For example, as part of a formal change impact analysis, a software engineer might need to specify which test cases to execute to verify that a defect report has been properly resolved. This example is close to the general definition of IR, "to find documents that satisfy an information need from within large collections". If the database of test cases contains overlapping test cases, it is possible that the engineer needs to report just one suitable test case. In this case precision is more important than recall, and it is fundamental that the tool presents few false positives among the top candidate trace links. Evaluating the performance of the IR-based trace recovery tool using constant cut points is suitable.

The second main use case of an IR-based trace recovery tool is to generate an entire set of trace links, i.e. a candidate traceability matrix. For instance, a traceability matrix needs to be established between n functional requirements and m test case descriptions during software evolution of a legacy system. If the number of artifacts is $n + m$, the number of possible trace links (i.e. the number of pair-wise comparisons needed) to consider is $n * m$, a number that quickly becomes infeasible for manual work. An engineer can in such cases use the output from an IR-based trace recovery tool as a starting point. The generation of a traceability matrix corresponds to running multiple simultaneous queries in a general IR system, and typically recall is favored over precision. There is no natural equivalent to this use case in the general IR domain. Furthermore, when generating an entire traceability matrix, it is improbable that the total number of correct trace links is known a priori, and consequently constant cut-points are less meaningful. A naïve cut-off strategy is to instead simply use a constant similarity threshold such as the cosine similarity 0.7. More promising cut-off strategies are based on variable thresholds or incremental approaches, as described in Section 2.2. Typically, the accuracies of traceability matrices generated by IR-based trace recovery tools are evaluated a posteriori, by analyzing how precision varies for different levels of recall.

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 for an SM (Kitchenham and Charters 2007; Petersen et al. 2008). 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 guidelines for secondary studies in software engineering (Kitchenham and Charters 2007), 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). 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[3] 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 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. (2000) and Marcus and Maletic (2003). However, we included

[3] www.zotero.org.

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

		Rationale/comments
	Inclusion criteria	
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 (De Lucia et al. 2006a). 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 (Ayari et al. 2007).
	Exclusion criteria	
E1	Answer is no to I1, I2 or I3	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 (Egyed and Grunbacher 2002; Spanoudakis et al. 2004), ontologies (Assawamekin et al. 2010), supervised machine learning (Spanoudakis et al. 2003), semantic networks (Lindvall et al. 2009), and dynamic analysis (Eisenbarth et al. 2003)
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	
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 (Runeson et al. 2007), detection of equivalent requirements (Falessi et al. 2010), and concept location (Marcus et al. 2004). We explicitly added the topics code clustering, class cohesion, and cross cutting concerns to clarify our scope.

publications describing all *independent replications* (deliberate variations of one or more major aspects), and *dependent replications* (same or very similar experimental setups) by other researchers (Shull et al. 2008).

- 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 (Di and Zhang 2009*).

- 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 (Kruchten 2004). Thus, we excluded linking approaches to entities such as e-mails (Bacchelli et al. 2010) and tacit knowledge (Stone and Sawyer 2006; Huffman Hayes et al. 2008).
- 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 (Van Rompaey and Demeyer 2009), and work linking source code to text expressed in specific modelling notation (Antoniol et al. 1999; Cleland-Huang et al. 2008).

4.2 Selection of Publications

The systematic identification of publications consisted of two main phases: (i) development of a gold standard of primary publications, and (ii) a search string that retrieves them, and a systematic search for publications, as shown in Fig. 3. 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 gold standard.[4] 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 gold 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 gold 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.

```
(traceability OR "requirements tracing" OR "requirements trace" OR
 "trace retrieval")
AND
(requirement* OR specif\/ication* 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)
```

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

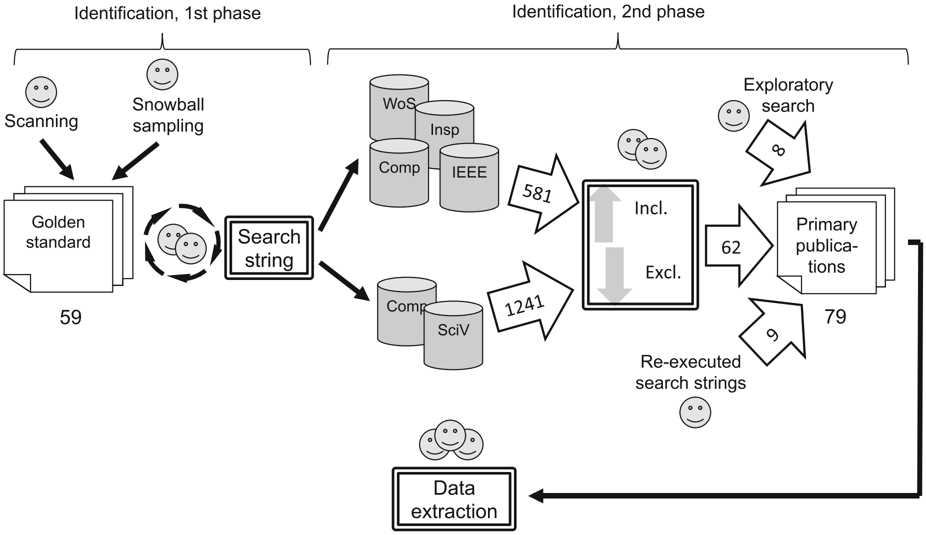


Fig. 3 Overview of the publication selection phase. *Smileys* show the number of people involved in a step, while double frames represent a validation. *Numbers* refer to number of publications

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 gold standard. The publications were filtered by our inclusion/exclusion criteria, as shown in Fig. 3, 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 (Randolph 2005), 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 1,000 search results. The inclusion/exclusion

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

	Search options	#Search results
Primary databases		
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		
ACM digital library	All fields, auto-stem	1,038
SciVerse hub beta	Science Direct+SCOPUS	203

criteria were then applied to the total 1,241 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 (Felizardo et al. 2011). REVIS takes a set of primary publications in an extended BibTeX format and, as presented in Fig. 4, visualizes the set as a document map (a), edge bundles (b), and a citation network for the document set (c). While REVIS was

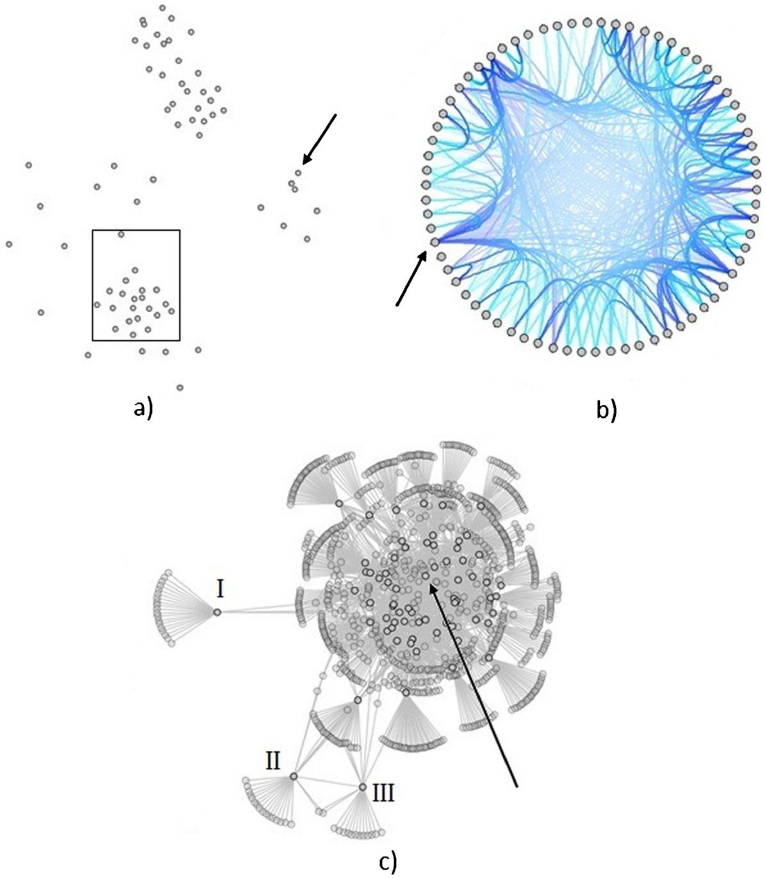


Fig. 4 Visualization of core primary publications. **a** Document map, shows similarities in language among the core primary publications. **b** Edge bundle, displays citations among the core primary publications. **c** Citation network, shows shared citations among the core primary publications

developed to support the entire mapping process, we solely used the tool as a means to visually validate our selection of publications.

In Fig. 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 (Antoniol et al. 2002*), the most cited publication on IR-based trace recovery. The closest publications in (a) are also authored by Antoniol et al. (1999*, 2000*). 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. (2004*, 2005*, 2006*a, 2007*, 2006*b, 2009*a, 2008*, 2009*b), Capobianco et al. (2009*b, *), and Oliveto et al. (2010*) 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. (2000*), describing work conducted concurrently with Antoniol et al. (2002*), 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. (2000*) 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. (2004*, 2006*). 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".

In addition to primary publications referred to elsewhere in the paper, they include Ali et al. (2011*a), Asuncion et al. (2010*), Cleland-Huang et al. (2005*, 2007*, 2010*), Czauderna et al. (2011*), Dekhtyar et al. (2011*), Di Penta et al. (2002*), Huffman Hayes et al. (2003*, 2005*), Klock et al. (2011*), Kong and Huffman Hayes (2011*), Kong et al. (2011*), Lormans et al. (2006*), Marcus et al. (2005*), McMillan et al. (2009*), Port et al. (2011*), Sundaram et al. (2005*), Winkler (2009*) and Yadla et al. (2005*).

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.

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 (Runeson et al. 2012). Particularly, we report deviations from the study guidelines (Kitchenham and Charters 2007).

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 (2007), 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.

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. (2007). 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.

External Validity refers to generalization from this study. In general, the external validity of a SM is strong, as the key idea is to aggregate as much as possible of the available literature. Also, our research scope is tight (cf. the inclusion/exclusion criteria in Table 3), and we do not claim that our map applies to other applications of IR in software engineering. Thus, the threats to external validity are minor. Furthermore, as the review protocol is presented in detail in Section 4, other researchers can judge the trustworthiness of the results in relation to the search strategy, inclusion/exclusion criteria, and the applied data extraction. Finally, *internal validity* concerns confounding factors that can affect the causal relationship between the treatment and the outcome. However, as our mapping study does not investigate casual relationships, and only relies on descriptive statistics, this threat is minimal.

5 Results

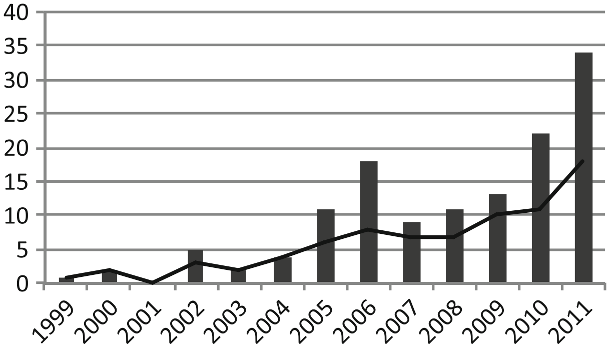
Following the method defined in Section 4.2, we identified 79 primary publications. Most of the publications were published in conferences or workshops (67 of 79, 85 %), while twelve (15 %) were published in scientific journals. Table 5 presents the top publication channels for IR-based trace recovery,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

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

Table 5 Top publication channels for IR-based trace recovery

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

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



5.1 IR Models Applied to Trace Recovery (RQ1)

In Fig. 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. For LSI, the dimensionality reductions applied in previous studies is reported in Appendix. Various probabilistic models have been applied in 29 of the 132 evaluations, including 14 applications of statistical LMs. Five of the applied approaches do not fit in the taxonomy; examples include utilizing swarm techniques (Sultanov and Huffman Hayes 2010*) and B-splines (Capobianco et al. 2009*b). As shown in Fig. 6, VSM is 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.

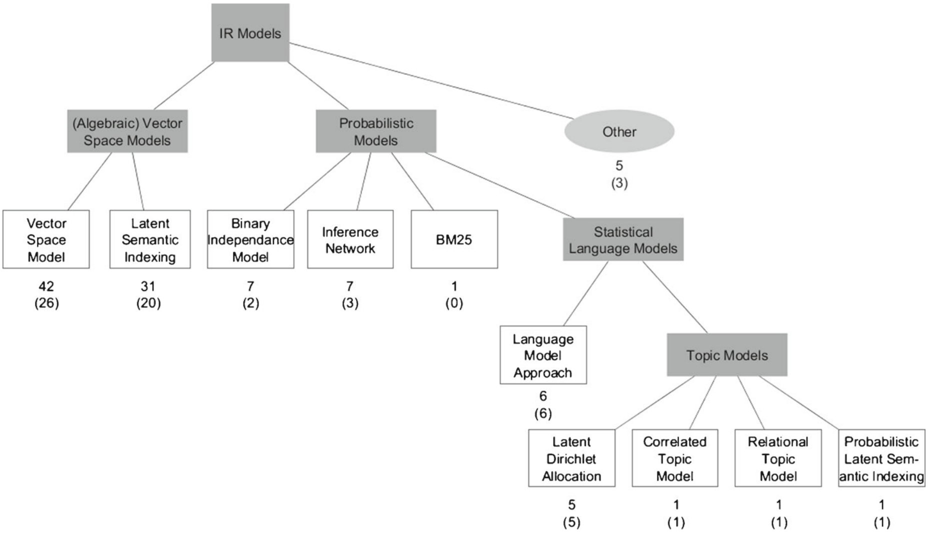


Fig. 6 Taxonomy of IR models in trace recovery. The *numbers* show in how many of the primary publications a specific model has been applied, the numbers in parentheses show IR models applied since 2008

Only 47 (72 %) 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 (e.g. Chen (2010*) and Kong et al. (2009*)), possibly due to page restriction. 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 (on the other hand some papers make a clear distinction, e.g. Wang et al. (2009*)). Among the publications reporting preprocessing, 32 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 Google Translate as a preprocessing step to translate NL text to English (Li et al. 2008*; Huffman Hayes et al. 2011*). 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 (Zhao et al. 2003*; Zhou and Yu 2007*), and one selects only nouns (Capobianco et al. 2009*b). Also, Capobianco et al. (2009*a) have explicitly explored the semantic role of nouns. 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 (Canfora and Cerulo 2006*; Abadi et al. 2008*; Ali et al. 2011*b).

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 (Salton and Buckley 1988), but since this level of detail often is omitted in publications on IR-based trace recovery, as also noted by Oliveto (2008), 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 (Mahmoud and Niu 2010*).

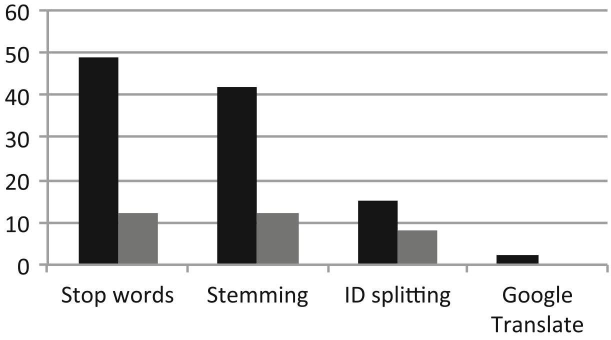


Fig. 7 Preprocessing operations used in IR-based trace recovery. The figure shows the number of times a specific operation has been reported in the primary publications. *Black bars* refer to preprocessing of NL text, *gray bars* show preprocessing of text extracted from source code

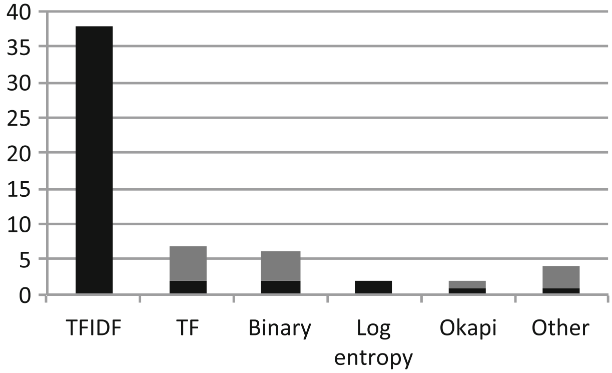


Fig. 8 Feature weighting schemes in IR-based trace recovery. *Bars* depict how many times a specific weighting scheme has been reported in the primary publications. *Black color* shows reported weighting in publications applying algebraic IR models

Several enhancement strategies to improve the performance of IR-based trace recovery tools are proposed, as presented in Fig. 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 (Jiang et al. 2008*). The most frequently applied enhancement strategy is relevance feedback, applied by e.g. De Lucia et al. (2006*b) and Huffman Hayes et al. (2007*), giving the human a chance to judge partial search results, followed by re-executing an improved search query. The following most frequently applied strategies, further described in Section 2.1, are applying a thesaurus to deal with synonyms, (e.g. proposed by Huffman Hayes et al. (2006*) and Leuser and Ott (2010*)), clustering results based on for instance document structure to improve presentation or ranking of recovered

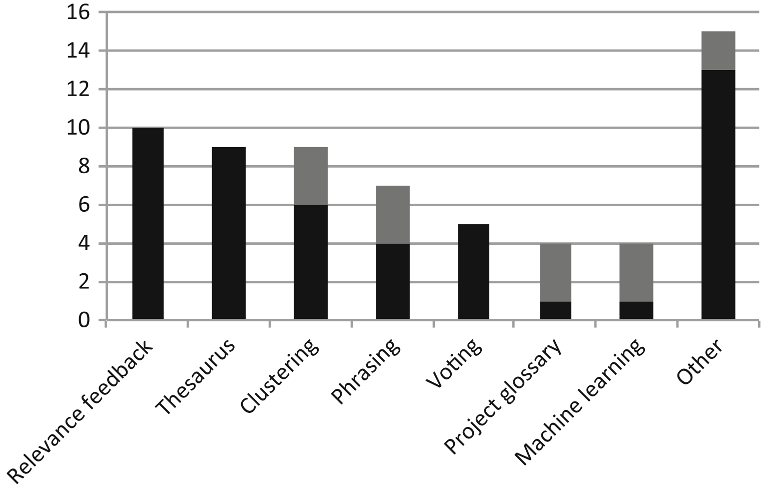
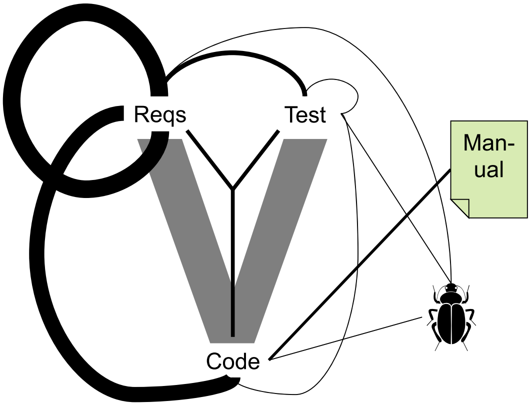


Fig. 9 Enhancement strategies in IR-based trace recovery. *Bars* depict how many times a specific strategy has been reported in the primary publications. *Black color* represents enhancements reported in publications using algebraic IR models

trace links, (explored by e.g. Duan and Cleland-Huang (2007*) and Zhou and Yu (2007*)), and phrasing, i.e. going beyond the BoW model by considering sequences of words, e.g. as described by Zou et al. (2006*) and Chen and Grundy (2011*). Other enhancement strategies repeatedly applied include: up-weighting terms considered important by applying a project glossary, e.g. (Zou et al. 2008*), machine learning approaches to improve results based on for example the existing trace link structure, e.g. (Di and Zhang 2009*), and combining the results from different retrieval models in voting systems, e.g. (Gethers et al. 2011*). Yet another set of enhancements have only been proposed in single primary publications, such as query expansion (Gibiec et al. 2010*), analyses of call graphs (Zhao et al. 2003*), regular expressions (Chen and Grundy 2011*), and smoothing filters (De Lucia et al. 2011*).

5.2 Types of Software Artifacts Linked (RQ2)

Figure 10 maps onto the classical software development V-model the various software artifact types that are 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 neither source nor target artifacts are dominated by NL text, results in fewer links between requirements-requirements, code-code, code-test, test-test and defect-defect than would have been the case if we had studied the entire field of IR applications within software engineering.



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

Fig. 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

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 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 (ten evaluations), links between requirements and tests (nine evaluations) and links between source code and manuals (six evaluations). Less frequently studied trace links include links between source code and defects/change requests (e.g. Gethers et al. 2011) and links between tests (Lormans et al. 2008*). In three primary publications, the types of artifacts traced are unclear, either not specified at all or merely described as 'documents' (e.g. Chen et al. 2011*).

5.3 Strength of Evidence (RQ3)

An overview of the datasets used for evaluations in the primary publications is shown in Fig. 11. In total we identified 132 evaluations; in 42 (32 %) cases proprietary artifacts were studied, either originating from development projects in private companies or the US agency NASA. 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 (e.g. Park et al. 2000*; Li et al. 2008*; Parvathy et al. 2008*). Figure 11 also depicts the sizes of the datasets used in the evaluations, wrt. the number of artifacts. 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 (Natt och Dag et al. 2004*) (now owned by Infor Global Solutions), and trace links between nine defect reports and 13,380 test cases at Research in Motion (Kaushik et al. 2011*).

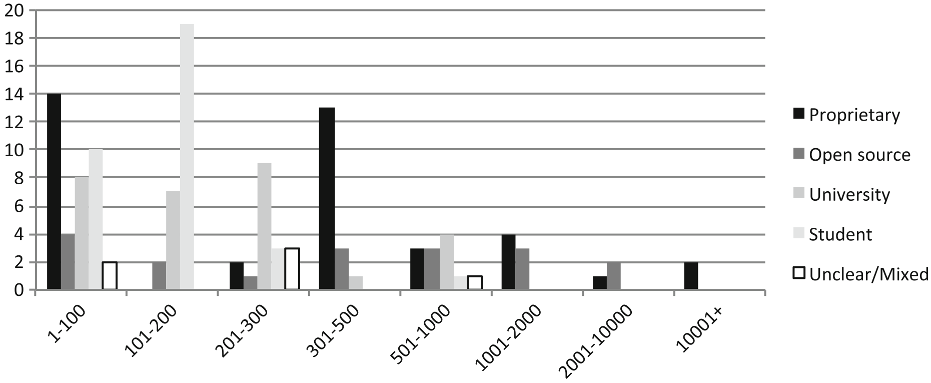


Fig. 11 Datasets used in studies on IR-based trace recovery. *Bars* show number and origin of artifacts

Table 6 Summary of the datasets most frequently used for evaluations

#	Dataset	Artifacts	Links	Origin	Development characteristics	Size[a]	Lang.
17	CM-1	Requirements specifying system requirements and detailed design	Bipartite dataset, many-to-many links	NASA	Embedded software development in governmental agency	455	English
16	EasyClinic	Use cases, sequence diagrams, source code, test case descriptions	Many-to-many links	Univ. of Salerno	Student project	150	Italian
8	MODIS	Requirements specifying system requirements and detailed design	Bipartite dataset, many-to-many links.	NASA	Embedded software development in governmental agency	68	English
7	Ice-breaker system (IBS)	Functional requirements and source code	Not publicly available in full detail	Robertson and Robertson (1999)	Textbook on requirements engineering	185	English
6	LEDA	Source code and user documentation	Bipartite dataset, many-to-one links	Max Planck Inst. for Informatics Saarbrücken	Scientific computing	296	English
5	Event-based traceability (EBT)	Functional requirements and source code	Not publicly available	DePaul Univ.	Tool from research project	138	English

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

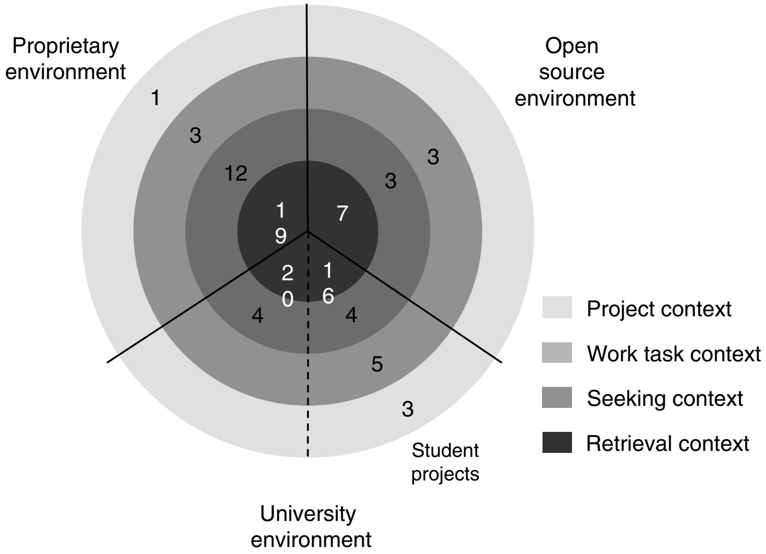


Fig. 12 Contexts of evaluations of IR-based trace recovery, along with study environments. *Numbers* show the number of primary publications that target each combination

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.[5] 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" (Huffman Hayes et al. 2006*). Fourteen publications (18 %) report results regarding the human analyst, two primary publications study both methods and human analysts (Antoniol et al. 2002*; De Lucia et al. 2006*a). 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" (Ingwersen and Järvelin 2005). As mentioned in Section 2, evaluations in the cave display an 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

[5] coest.org.

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 (Li et al. 2008*).

6 Discussion

This section discusses the results reported in the previous section and concludes on the 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 6.4, we map our recommendations to the traceability challenges articulated by CoEST (Gotel et al. 2012).

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 (Salton et al. 1975) and LSI, the extension developed by Deerswester in the 90s (Deerwester et al. 1990). Also, we show that VSM has been implemented more frequently than LSI, in contrast to what was reported by Binkley and Lawrie (2010). 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 (Zhai 2007), LMs (Ponte and Croft 1998) have been getting more attention in the last years. Regarding enhancements strategies, relevance feedback, introduction of a thesaurus and clustering of results are the most frequently applied.

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 (Borg et al. 2012a). Thus, omitting details in the reporting obstructs replications and the possibility to advance the field of trace recovery through secondary studies and evidence-based software engineering techniques (Jedlitschka et al. 2008). 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. (2012) dedicated an entire chapter of their recent book to this issue. However, we hope that Section 2.1 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 Fig. 6, and use general traceability terminology as proposed by Cleland-Huang et al. (2012).
- Authors of articles on IR-based trace recovery should carefully report the implemented IR model, including the features considered, to enable aggregating empirical evidence.
- Technology-oriented experiments on IR-based trace recovery should adhere to rigorous methodologies such as the evaluation framework by Huffman Hayes and Dekhtyar (2005a).

6.2 Types of Software Artifacts Linked (RQ2)

Most published evaluations on IR-based trace recovery aim at establishing trace links between requirements in a wide sense, 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, as it is considered the "starting point" of development. 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 (Sabaliauskaite et al. 2010). 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 (De Lucia et al. 2004*, 2006*a, *b, 2008*, 2009*a, *b, 2011*), 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, to explore recovery of different types of trace links.
- Studies should go beyond bipartite datasets to better represent the heterogeneous information landscape of software engineering, thus enabling studies on several types of links within the same datasets.

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 (Scacchi 2002; Canfora and Cerulo 2006*), 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 (Natt och Dag et al. 2004*; Kaushik et al. 2011*). As a result, the question of whether the state-of-the-art IR-based trace recovery scales to larger document spaces or not remains unanswered. In the empirical NLP community, Banko and Brill (2001) showed that some conclusions (related to machine learning techniques for NL disambiguation) drawn on small datasets may not carry over to very large datasets. Researchers on IR-based trace recovery appear to be aware of the scalability issue however, as it is commonly mentioned as a threat to external validity and suggested as future work in the primary publications (Huffman Hayes et al. 2004*; De Lucia et al. 2007*; Leuser 2009*; Wang et al. 2009*; Gibiec et al. 2010*; Mahmoud and Niu 2011*). On the other hand, one reason for the many studies on small datasets is the challenge involved in obtaining the complete set of correct trace links, i.e. a gold standard or ground truth, required for evaluations. In certain domains, e.g. development of safety-critical systems, such information might already be available. If such information is missing however, a traceability researcher first needs to establish the gold standard, which requires much work for a large dataset.

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 (Smeaton and Harman 1997), but on the other hand it can also lead the research community to over-engineering tools on specific datasets (Borg et al. 2012a). Thus, the community needs to consider the risk of optimization against those specific benchmarks, which may make the final result less suitable in the general case, if the benchmarks are not representative enough. The benchmark discussion has been very active in the traceability community the last years (Dekhtyar and Huffman Hayes 2006; Dekhtyar et al. 2007; Cleland-Huang et al. 2011; Ben Charrada et al. 2011*; Gotel et al. 2012).

A related problem, in particular for proprietary datasets that cannot be disclosed, is that datasets often are poorly described (Borg et al. 2012b). 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, providing information about the datasets and their contexts is also important for interpreting results and their validity, in line with previous work by Ali et al. (2012) and Borg et al. (2012b). For example, researchers should report as much of the industrial context from which the dataset arose as encouraged by Kitchenham et al. (2002). As a starting point, researchers could use the preliminary framework for describing industrial context by Petersen and Wohlin (2009).

As discussed in Section 3.4, P-R values can be reported from IR-based trace recovery evaluations in different ways. Unfortunately, the reported values are not always properly explained in the primary publications. In the evaluation report, it is central to state whether a query-based or matrix-based evaluation style has been used, as well as which cut-off strategies were applied. Furthermore, for query-based evaluations (closer resembling traditional IR), we agree with the opinion of Spärck Jones et al. (2000), that reporting only precision at standard recall levels is opaque. The figures obscure the actual numbers of retrieved documents needed to get beyond low recall, and should be complemented by P-R values from a constant cut-point cut-off strategy. Moreover, regarding both query-based and matrix-based evaluation styles, reporting also secondary measures (such as MAP and DCG) is a step toward more mature evaluations.

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 (Ingwersen and Järvelin 2005). 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. (Huffman Hayes et al. 2004*; Huffman Hayes and Dekhtyar 2005b; Dekhtyar et al. 2007*a; Cuddeback et al. 2010*) and De Lucia et al. (2006*b, 2008*, 2009*a). 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 (Li et al. 2008*) and three evaluations in student projects (De Lucia et al. 2005*, 2006*a, 2007*) 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 both the context and the datasets used in evaluations, in particular when they cannot be disclosed for confidentiality reasons.
- P-R values should be complemented by secondary measures such as MAP and DCG, and it should be made clear whether a query-based or matrix-based evaluation style was used.
- Focus on tool enhancements "in the cave" should be shifted towards evaluations in the work task or project context.

6.4 In the Light of the CoEST Research Agenda

Gotel et al. (2012) recently published a framework of challenges in traceability research, a CoEST community effort based on a draft from 2006 (Cleland-Huang et al. 2006). 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 to be solved in 2035, as presented in Table 7. Our work mainly

Table 7 Traceability research themes defined by CoEST (Gotel et al. 2012). Ubiquitous traceability is referred to as "the grand challenge of traceability", since it requires significant progress in the other research themes

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 visualizations, to handle very large datasets and the longevity of these data
Trusted traceability	To develop cost-benefit models for analyzing 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

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 terminology, methods, practices and the like. 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 evaluation". To guide future empirical studies, we propose an adapted version of the model of IR evaluation contexts by Ingwersen and Järvelin (2005), 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 by 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 (Kitchenham and Charters 2007). Our study constitutes the most extensive summary of publications of IR-based trace recovery yet published.

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 (Fig. 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 levels 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 reported in Fig. 12. Instead, 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. Thus, we argue that industrial in-vivo evaluations are needed to motivate the feasibility of the approach and further studies on the topic, in which IR-based trace recovery should be studied within the full complexity of an industrial setting. As such, our empirical findings intensify the recent call for additional empirical studies by CoEST (Gotel et al. 2012).

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 secondary studies on candidate trace link accuracies challenging. We argue that both the standard measures precision at fixed recall levels and P-R at specific document cut-offs should be reported when applicable, complemented by secondary measures such as MAP and DCG.

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. Another option for the future is to aggregate results from the innermost evaluation context, as P-R values repeatedly have been reported in the primary studies. However, such a secondary study must be carefully designed to allow a valid synthesis across different studies. Finally, future work could include other mapping dimensions, such as categorizing the primary publications according to other frameworks, e.g. positioning them related to the CoEST research themes.

Acknowledgements This work was funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering.[6] Thanks go to our librarian Mats Berglund for working on the search strings, and Lorand Dali for excellent comments on IR details.

Appendix: Classification of Primary Publications

Table 8 presents 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 (2003) (354 citations) and Antoniol et al. (2000) (85 citations) are not listed. Applied IR models are reported in the fourth column. For LSI, the number of dimensions (k) in the reduced term-document space is reported in parenthesis, divided per dataset when possible. The number of dimensions is reported either as a fixed number of dimensions, an interval of dimensions, a dimensionality reduction in percent, or 'N/A' when the information is not available. A bold number represents that the best choice, as concluded by the original authors. Regarding LDA, the number of topics (t) is reported. 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, i.e. the total number of artifacts in the dataset, 'N/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, Table 9 shows the distinctly most productive authors and affiliations, based upon our primary publications.

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

Table 8 Classification of 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 (k=10 (MODIS), 100 (CM-1))	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 software artifact management systems using information retrieval methods	De Lucia, Fasano, Oliveto, Tortora	LSI (k=30–100%)	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 (N/A), 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 (k=20%)	Ind: Philips (359), Stud: PacMan (46), Callisto (N/A)	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 (k=10–50%)	Stud: EasyClinic (150)	Level 1
58	Recovery of traceability links between software documentation and source code	Marcus, Maletic, Sergeyev	LSI (N/A)	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 (N/A)	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 (N/A), LM, LDA (t=50–300)	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 (k=10, **19**, (MODIS), 60 (EasyClinic))	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

Table 8 (continued)

Cit.	Title	Authors	IR mod.	Dataset	Evaluation
30	Assessing IR-based traceability recovery tools through controlled experiments	De Lucia, Oliveto, Tortora	LSI (N/A)	Stud: EasyClinic (150)	Level 3 (20, 12 subj.)
29	A traceability technique for specifications	Abadi, Nisenson, Simionovici	VSM, LSI (k=5–100, **16** (SCA), **96** (CORBA)), PLSI (k=5–128), SDR (k=5–128), LM	OS: SCA (1311), CORBA (3340)	Level 2
29	Can information retrieval techniques effectively support traceability link recovery?	De Lucia, Fasano, Oliveto, Tortora	LSI (k=20%)	Stud: EasyClinic (150), Univ: ADAMS (309), LEDA (803)	Level 1, Level 4 (150 subj.)
29	Software traceability with topic modeling	Asuncion, Asuncion, Taylor	LSI (k=10), LDA (t=10, 20,30)	Univ: ArchStudio (N/A), Stud: EasyClinic (160) Stud: EasyClinic (160)	Level 1
29	Speeding up requirements to management in a product software company: linking customer wishes 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, Sundaram, Holbrook, 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
21	Combining textual and structural analysis of software artifacts for traceability link recovery	McMillan, Poshyvanyk, Revelle	LSI (k=15, 25, 50, **75**)	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.)

Table 8 (continued)

Cit.	Title	Authors	IR mod.	Dataset	Evaluation
18	Incremental latent semantic indexing for automatic traceability link evolution management	Jiang, Nguyen, Chen, Jaygarl, Chang	LSI (k=10%)	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 (N/A)	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 (k=**10**,19,29 (MODIS), 100,200 (CM-1))	Ind: CM-1 (455), MODIS (68)	Level 2
11	Challenges for semi-automatic trace recovery in the automotive domain	Leuser	VSM, LSI (N/A)	Ind: Daimler AG (1500)	Level 1
11	Monitoring requirements coverage using reconstructed views: an industrial case study	Lormans, Gross, van Deursen, Stehouwer, van Solingen	LSI (N/A)	Ind: LogicaCMG (219)	Level 1
11	On the role of the nouns in IR-based traceability recovery	Capobianco, De Lucia, Oliveto, Panichella, Panichella	LSI (N/A), 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 (k=40%)	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 charecteristics of software engineering	Wang, Lai, Liu	LSI (N/A), BIM	Univ: LEDA (597), Univ: IBS (270)	Level 1
8	Trace retrieval for evolving artifacts	Winkler	LSI (k=15%)	Ind: Robert Bosch GmbH (500), MODIS (68)	Level 1
8	Traceability recovery using numerical analysis	Capobianco, De Lucia, Oliveto, Panichella, Panichella	VSM, LSI (N/A), LM, B-splines	Stud: EasyClinic (150)	Level 1
7	Assessing traceability of software engineering artifacts	Sundaram, Huffman Hayes, Dekhtyar, Holbrook	VSM, LSI (k=10,25, 30,40,60 (MODIS), 10,25,100,200, 400 (CM-1), 5,10,15,25,40 (Waterloo)	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 8 (continued)

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 (N/A), BIM LDA (N/A), 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 (N/A), uDig 1.1.1 (N/A)	Level 1
2	Source code indexing for automated tracing	Mahmoud, Niu	VSM	Stud: eTour (174), iTrust (264)	Level 1
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, Guéhéneuc, 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 (N/A)	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 (N/A)	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 (N/A)	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

Table 8 (continued)

Cit.	Title	Authors	IR mod.	Dataset	Evaluation
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 (N/A)	Level 1
0	A comparative study of document correlation techniques for traceability analysis	Parvathy, Vasudevan, Balakrishnan	VSM, LSI (k=10), LDA (t=21), 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 (k=50–500, **150–200**)	Ind: RIM (13389)	Level 1
0	Requirements traceability for object oriented systems by partitioning source code	Ali, Guéhéneuc, 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 (N/A)	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 9 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

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

References

Abadi A, Nisenson M, Simionovici Y (2008*) A traceability technique for specifications. In: Proceedings of the 16th international conference on program comprehension, pp 103–112

Aitchison J, Bawden D, Gilchrist A (2000) Thesaurus construction and use: a practical manual, 4th edn. Routledge

Ali N, Guéhéneuc Y, Antoniol G (2011*a) Requirements traceability for object oriented systems by partitioning source code. In: Proceedings of the 18th working conference on reverse engineering, pp 45–54

Ali N, Guéhéneuc Y, Antoniol G (2011*b) Trust-Based requirements traceability. In: Proceedings of the 19th international conference on program comprehension, pp 111–120

Ali N, Guéhéneuc Y, Antoniol G (2012) Factors impacting the inputs of traceability recovery approaches. In: Cleland-Huang J, Gotel O, Zisman A (eds) Software and systems traceability, Springer

Antoniol G, Potrich A, Tonella P, Fiutem R (1999) Evolving object oriented design to improve code traceability. In: Proceedings of the 7th international workshop on program comprehension, pp 151–160

Antoniol G, Canfora G, De Lucia A, Merlo E (1999*) Recovering code to documentation links in OO systems. In: Proceedings of the 6th working conference on reverse engineering, pp 136–144

Antoniol G, Canfora G, Casazza G, De Lucia A (2000) Information retrieval models for recovering traceability links between code and documentation. In: Conference on software maintenance, pp 40–49

Antoniol G, Canfora G, Casazza G, De Lucia A, Merlo E (2000*) Tracing object-oriented code into functional requirements. In: Proceedings of the 8th international workshop on program comprehension, pp 79–86

Antoniol G, Canfora G, Casazza G, De Lucia A, Merlo E (2002*) Recovering traceability links between code and documentation. In: Transactions on software engineering, vol 28, pp 970–983

Assawamekin N, Sunetnanta T, Pluempitiwiriyawej C (2010) Ontology-based multiperspective requirements traceability framework. Knowl Inf Syst 25(3):493–522

Asuncion H, Asuncion A, Taylor R (2010*) Software traceability with topic modeling. In: Proceedings of the international conference on software engineering, pp 95–104

Ayari K, Meshkinfam P, Antoniol G, Di Penta M (2007) 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, pp 215–228

Bacchelli A, Lanza M, Robbes R (2010) Linking e-mails and source code artifacts. In: Proceedings of the 32nd international conference on software engineering, pp 375–384

Baeza-Yates R, Ribeiro-Neto B (2011) Modern information retrieval: the concepts and technology behind search. Addison-Wesley

Banko M, Brill E (2001) Scaling to very very large corpora for natural language disambiguation. In: Proceedings of the 39th annual meeting on association for computational linguistics, pp 26–33

Ben Charrada E, Caspar D, Jeanneret C, Glinz M (2011*) Towards a benchmark for traceability. In: Proceedings of the 12th international workshop on principles on Software evolution, pp 21–30

Bianchi A, Fasolino A, Visaggio G (2000) An exploratory case study of the maintenance effectiveness of traceability models. In: Proceedings of the 8th international workshop on program comprehension, pp 149–158

Binkley D, Lawrie D (2010) Information retrieval applications in software maintenance and evolution. In: Marciniak J (ed) Encyclopedia of software engineering, 2nd edn, Taylor & Francis

Blei D, Lafferty J (2007) A correlated topic model of science. Ann Appl Stat 1(1):17–35

Blei D, Ng A, Jordan M (2003) Latent dirichlet allocation. J Mach Learn Res 3(4–5):993–1022

Borg M, Pfahl D (2011*) Do better IR tools improve the accuracy of engineers' traceability recovery? In: Proceedings of the international workshop on machine learning technologies in software engineering, pp 27–34

Borg M, Runeson P, Brodén L (2012a) 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

Borg M, Wnuk K, Pfahl D (2012b) Industrial comparability of student artifacts in traceability recovery research - an exploratory survey. In: Proceedings of the 16th european conference on software maintenance and reengineering

Borillo M, Borillo A, Castell N, Latour D, Toussaint Y, Felisa Verdejo M (1992) Applying linguistic engineering to spatial software engineering: the traceability problem. In: Proceedings of the 10th european conference on artificial intelligence, pp 593–595

Bras M, Toussaint Y (1993) Artificial intelligence tools for software engineering: Processing natural language requirements. In: Applications of artificial intelligence in engineering, pp 275–290

Brereton P, Kitchenham B, Budgen D, Turner M, Khalil M (2007) Lessons from applying the systematic literature review process within the software engineering domain. J Syst Software 80(4):571–583

Canfora G, Cerulo L (2006*) Fine grained indexing of software repositories to support impact analysis. In: Proceedings of the international workshop on mining software repositories, pp 105–111

Capobianco G, De Lucia A, Oliveto R, Panichella A, Panichella S (2009*a) On the role of the nouns in IR-based traceability recovery. In: Proceedings of the 17th international conference on program comprehension, pp 148–157

Capobianco G, De Lucia A, Oliveto R, Panichella A, Panichella S (2009*b) Traceability recovery using numerical analysis. In: Proceedings of the 16th working conference on reverse engineering, pp 195–204

Carnegie Mellon Software Engineering Institute (2010) CMMI for development, version 1.3

Castell N, Slavkova O, Toussaint Y, Tuells A (1994) 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, pp 37–44

Chang J, Blei D (2010) Hierarchical relational models for document networks. Ann Appl Stat 4(1):124–150

Charikar M, Chekuri C, Feder T, Motwani R (1997) Incremental clustering and dynamic information retrieval. In: Proceedings of the 29th annual ACM symposium on theory of computing, pp 626–635

Chen X (2010*) Extraction and visualization of traceability relationships between documents and source code. In: Proceedings of the international conference on automated software engineering, pp 505–509

Chen X, Grundy J (2011*) Improving automated documentation to code traceability by combining retrieval techniques. In: Proceedings of the 26th international conference on automated software engineering, pp 223–232

Chen X, Hosking J, Grundy J (2011*) A combination approach for enhancing automated traceability. In: Proceeding of the 33rd international conference on software engineering, (NIER track), pp 912–915

Cleland-Huang J, Chang CK, Christensen M (2003) Event-based traceability for managing evolutionary change. Trans Software Eng 29(9):796–810

Cleland-Huang J, Settimi R, Duan C, Zou XC (2005*) Utilizing supporting evidence to improve dynamic requirements traceability. In: Proceedings of the 13th international conference on requirements engineering, pp 135–144

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

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

Cleland-Huang J, Marrero W, Berenbach B (2008) Goal-Centric traceability: Using virtual plumblines to maintain critical systemic qualities. Trans Software Eng 34(5):685–699

Cleland-Huang J, Czauderna A, Gibiec M, Emenecker J (2010*) A machine learning approach for tracing regulatory codes to product specific requirements. In: Proceedings international conference on software engineering, pp 155–164

Cleland-Huang J, Czauderna A, Dekhtyar A, Gotel O, Huffman Hayes J, Keenan E, Maletic J, Poshyvanyk D, Shin Y, Zisman A, Antoniol G, Berenbach B, Egyed A, Maeder P (2011) 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

Cleland-Huang J, Gotel O, Zisman A (eds) (2012) Software and systems traceability. Springer

Cleverdon C (1991) 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, pp 3–12

Croft B, Turtle H, Lewis D (1991) The use of phrases and structured queries in information retrieval. In: Proceedings of the 14th annual international ACM SIGIR conference on research and development in information retrieval, pp 32–45

Cuddeback D, Dekhtyar A, Huffman Hayes J (2010*) Automated requirements traceability: the study of human analysts. In: Proceedings of the 18th international requirements engineering conference, pp 231–240

Czauderna A, Gibiec M, Leach G, Li Y, Shin Y, Keenan E, Cleland-Huang J (2011*) 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, pp 75–78

De Lucia A, Fasano F, Oliveto R, Tortora G (2004*) Enhancing an artefact management system with traceability recovery features. In: Proceedings of the 20th international conference on software maintenance, pp 306–315

De Lucia A, Fasano F, Oliveto R, Tortora G (2005*) ADAMS re-trace: A traceability recovery tool. In: Proceedings of the 9th European conference on software maintenance and reengineering, pp 32–41

De Lucia A, Di Penta M, Oliveto R, Zurolo F (2006a) COCONUT: COde COmprehension nurturant using traceability. In: Proceedings of the 22nd international conference on software maintenance, pp 274–275

De Lucia A, Di Penta M, Oliveto R, Zurolo F (2006b) Improving comprehensibility of source code via traceability information: A controlled experiment. In: Proceedings of the 14th international conference on program comprehension, pp 317–326

De Lucia A, Fasano F, Oliveto R, Tortora G (2006*a) Can information retrieval techniques effectively support traceability link recovery? In: Proceedings of the 14th international conference on program comprehension, pp 307–316

De Lucia A, Oliveto R, Sgueglia P (2006*b) Incremental approach and user feedbacks: A silver bullet for traceability recovery? In: Proceedings of the international conference on software maintenance, pp 299–308

De Lucia A, Fasano F, Oliveto R, Tortora G (2007*) Recovering traceability links in software artifact management systems using information retrieval methods. Trans Softw Eng Methodol 16(4)

De Lucia A, Fasano F, Oliveto R (2008) Traceability management for impact analysis. In: Frontiers of software maintenance, pp 21–30

De Lucia A, Oliveto R, Tortora G (2008*) 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, pp 39–48

De Lucia A, Oliveto R, Tortora G (2009*a) Assessing IR-based traceability recovery tools through controlled experiments. Empir Software Eng 14(1):57–92

De Lucia A, Oliveto R, Tortora G (2009*b) The role of the coverage analysis during IR-based traceability recovery: a controlled experiment. In: Proceedings of the 25th international conference on software maintenance, pp 371–380

De Lucia A, Di Penta M, Oliveto R, Panichella A, Panichella S (2011*) Improving IR-based traceability recovery using smoothing filters. In: Proceedings of the 19th international conference on program comprehension, pp 21–30

De Lucia A, Marcus A, Oliveto R, Poshyvanyk D (2012) Information retrieval methods for automated traceability recovery. In: Cleland-Huang J, Gotel O, Zisman A (eds) Software and systems traceability, Springer

Deerwester S, Dumais S, Furnas G, Landauer T, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391–407

Dekhtyar A, Huffman Hayes J (2006) Good benchmarks are hard to find: Toward the benchmark for information retrieval applications in software engineering. In: Proceedings of the 22th international conference on software maintenance

Dekhtyar A, Huffman Hayes J, Antoniol G (2007) Benchmarks for traceability? In: Proceedings of the international symposium on grand challenges in traceability

Dekhtyar A, Huffman Hayes J, Larsen J (2007*a) 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

Dekhtyar A, Huffman Hayes J, Sundaram S, Holbrook A, Dekhtyar O (2007*b) Technique integration for requirements assessment. In: Proceedings of the 15th international requirements engineering conference, pp 141–152

Dekhtyar A, Dekhtyar O, Holden J, Huffman Hayes J, Cuddeback D, Kong W (2011*) On human analyst performance in assisted requirements tracing: statistical analysis. In: Proceedings of the 19th international requirements engineering conference, pp 111–120

Di F, Zhang M (2009*) An improving approach for recovering requirements-to-design traceability links. In: Proceedings of the international conference on computational intelligence and software engineering, pp 1–6

Di Penta M, Gradara S, Antoniol G (2002*) Traceability recovery in RAD software systems. In: Proceedings of the 10th international workshop on program comprehension, pp 207–216

Dit B, Revelle M, Gethers M, Poshyvanyk D (2011) Feature location in source code: a taxonomy and survey. J Softw Maint Evol 25(1):53–95

Dömges R, Pohl K (1998) Adapting traceability environments to project-specific needs. Commun ACM 41(12):54–62

Duan C, Cleland-Huang J (2007*) Clustering support for automated tracing. In: Proceedings of the international conference on automated software engineering, pp 244–253

Egyed A, Grunbacher P (2002) Automating requirements traceability: beyond the record replay paradigm. In: Proceedings of the 17th international conference on automated software engineering, pp 163–171

Eisenbarth T, Koschke R, Simon D (2003) Locating features in source code. Trans Software Eng 29(3):210– 224

Falessi D, Cantone G, Canfora G (2010) A comprehensive characterization of NLP techniques for identifying equivalent requirements. In: Proceedings of the 4th international symposium on empirical software engineering and measurement

Felizardo KR, Salleh N, Martins RM, Mendes E, MacDonell SG, Maldonado JC (2011) Using visual text mining to support the study selection activity in systematic literature reviews. In: Proceedings of the 5th international symposium on empirical software engineering and measurement, pp 77–86

Fiutem R, Antoniol G (1998) Identifying design-code inconsistencies in object-oriented software: a case study. In: Proceedings of the international conference on software maintenance, pp 94–102

Gay G, Haiduc S, Marcus A, Menzies T (2009) On the use of relevance feedback in IR-based concept location. In: Proceedings of the 25th international conference on software maintenance, pp 351–360

Gethers M, Kagdi H, Dit B, Poshyvanyk D (2011) An adaptive approach to impact analysis from change requests to source code. In: Proceedings of the 26th international conference on automated software engineering, pp 540–543

Gethers M, Oliveto R, Poshyvanyk D, De Lucia A (2011*) On integrating orthogonal information retrieval methods to improve traceability recovery. In: Proceedings of the 27th international conference on software maintenance, pp 133–142

Gibiec M, Czauderna A, Cleland-Huang J (2010*) Towards mining replacement queries for hard-to-retrieve traces. In: Proceedings of the international conference on automated software engineering, pp 245–254

Gotel O, Finkelstein C (1994) An analysis of the requirements traceability problem. In: Proceedings of the first international conference on requirements engineering, pp 94–101

Gotel O, Cleland-Huang J, Huffman Hayes J, Zisman A, Egyed A, Grünbacher P, Dekhtyar A, Antoniol G, Maletic J (2012) The grand challenge of traceability (v1.0). In: Cleland-Huang J, Gotel O, Zisman A (eds) Software and systems traceability, Springer

Heindl M, Biffl S (2005) 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, pp 60–69

Hofman T (2001) Unsupervised learning by probabilistic latent semantic analysis. Mach Learn 42(1–2):177–196

Huffman Hayes J, Dekhtyar A (2005a) A framework for comparing requirements tracing experiments. Int J Softw Eng Knowl Eng 15(5):751–781

Huffman Hayes J, Dekhtyar A (2005b) 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, pp 20–23

Huffman Hayes J, Dekhtyar A, Osborne J (2003*) Improving requirements tracing via information retrieval. In: Proceedings of the 11th international requirements engineering conference, pp 138–147

Huffman Hayes J, Dekhtyar A, Sundaram S, Howard S (2004*) Helping analysts trace requirements: An objective look. In: Proceedings of the 12th international conference on requirements engineering, pp 249–259

Huffman Hayes J, Dekhtyar A, Sundaram S (2005*) Text mining for software engineering: how analyst feedback impacts final results. In: Proceedings of the international workshop on mining software repositories, pp 1–5

Huffman Hayes J, Dekhtyar A, Sundaram S (2006*) Advancing candidate link generation for requirements tracing: the study of methods. Trans Softw Eng 32(1):4–19

Huffman Hayes J, Dekhtyar A, Sundaram S, Holbrook A, Vadlamudi S, April A (2007*) REquirements TRacing on target (RETRO): improving software maintenance through traceability recovery. Innov Syst Softw Eng 3(3):193–202

Huffman Hayes J, Antoniol G, Guéhéneuc Y (2008) PREREQIR: recovering Pre-Requirements via cluster analysis. In: Proceedings of the 15th working conference on reverse engineering, pp 165–174

Huffman Hayes J, Sultanov H, Kong W, Li W (2011*) 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, ACM, pp 50–53

Ingwersen P, Järvelin K (2005) The turn: integration of information seeking and retrieval in context. Springer

International Electrotechnical Commission (2003) IEC 61511-1 ed 1.0, safety instrumented systems for the process industry sector

International Organization for Standardization (2011) ISO 26262-1:2011 road vehicles – functional safety –

Järvelin K, Kekäläinen J (2000) 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, pp 41–48

Jedlitschka A, Ciolkowski M, Pfahl D (2008) Reporting experiments in software engineering. In: Shull F, Singer J, Sjoberg D (eds) Guide to advanced empirical software engineering, Springer, London, pp 201–228

Jiang H, Nguyen T, Chen I, Jaygarl H, Chang C (2008*) Incremental latent semantic indexing for automatic traceability link evolution management. In: Proceedings of the 23rd international conference on automated software engineering, pp 59–68

Katta V, Stålhane T (2011) A conceptual model of traceability for safety systems. In: Proceedings of the complex systems design & management conference

Kaushik N, Tahvildari L, Moore M (2011*) Reconstructing traceability between bugs and test cases: an experimental study. In: Proceedings of the 18th working conference on reverse engineering, pp 411–414

Kekäläinen J, Järvelin K (2002) Evaluating information retrieval systems under the challenges of interaction and multidimensional dynamic relevance. In: Proceedings of the COLIS 4 conference pp 253–270

Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report

Kitchenham B, Pfleeger S, Pickard L, Jones P, Hoaglin D, El Emam K, Rosenberg J (2002) Preliminary guidelines for empirical research in software engineering. Trans Softw Eng Methodol 28(8):721–734

Kitchenham B, Budgen D, Brereton P (2011) Using mapping studies as the basis for further research—a participant-observer case study. Inform Softw Technol 53(6):638–651

Klock S, Gethers M, Dit B, Poshyvanyk D (2011*) Traceclipse: an eclipse plug-in for traceability link recovery and management. In: Proceedings of the 6th international workshop on traceability in emerging forms of software engineering, pp 24–30

Kong L, Li J, Li Y, Yang Y, Wang Q (2009*) A requirement traceability refinement method based on relevance feedback. In: Proceedings of the 21st international conference on software engineering and knowledge engineering

Kong W, Huffman Hayes J (2011*) Proximity-based traceability: an empirical validation using ranked retrieval and set-based measures. In: Proceedings of the 1st international workshop on empirical requirements engineering, pp 45–52

Kong W, Huffman Hayes J, Dekhtyar A, Holden J (2011*) How do we trace requirements: an initial study of analyst behavior in trace validation tasks. In: Proceeding of the 4th international workshop on cooperative and human aspects of software engineering, pp 32–39

Kruchten P (2004) The rational unified process: an introduction. Addison-Wesley Professional

Leuser J (2009*) Challenges for semi-automatic trace recovery in the automotive domain. In: Proceedings of the international workshop on traceability in emerging forms of software engineering, pp 31–35

Leuser J, Ott D (2010*) Tackling semi-automatic trace recovery for large specifications. In: Requirements engineering: foundation for software quality, pp 203–217

Lewis D (1998) Naive (Bayes) at forty: The independence assumption in information retrieval. In: Machine learning: ECML-98, vol 1398, Springer, pp 4–15

Li Y, Li J, Yang Y, Li M (2008*) Requirement-centric traceability for change impact analysis: a case study. In: International conference on software process, pp 100–111

Liddy E (2001) Natural language processing, 2nd edn. Encyclopedia of Library and Information Science, Marcel Decker

Lin J, Chan L, Cleland-Huang J, Settimi R, Amaya J, Bedford G, Berenbach B, Khadra OB, Chuan D, Zou X (2006) Poirot: A distributed tool supporting Enterprise-Wide automated traceability. In: Proceedings of the 14th international conference on requirements engineering, pp 363–364

Lindvall M, Feldmann R, Karabatis G, Chen Z, Janeja V (2009) Searching for relevant software change artifacts using semantic networks. In: Proceedings of the symposium on applied computing, pp 496–500

Lormans M, van Deursen A (2006*) Can LSI help reconstructing requirements traceability in design and test? In: Proceedings of the 10th European conference on software maintenance and reengineering, pp 45–54

Lormans M, Gross H, van Deursen A, van Solingen R, Stehouwer A (2006*) Monitoring requirements coverage using reconstructed views: An industrial case study. In: Procedings of the 13th working conference on reverse engineering, pp 275–284

Lormans M, Van Deursen A, Gross H (2008*) An industrial case study in reconstructing requirements views. Empir Software Eng 13(6):727–760

Mahmoud A, Niu N (2010*) Using semantics-enabled information retrieval in requirements tracing: An ongoing experimental investigation. In: Proceedings of the international computer software and applications conference, pp 246–247

Mahmoud A, Niu N (2011*) Source code indexing for automated tracing. In: Proceeding of the 6th international workshop on traceability in emerging forms of software engineering, pp 3–9

Manning C, Raghavan P, Schutze H (2008) Introduction to information retrieval. Cambridge University Press

Marcus A, Maletic J (2003) Recovering documentation-to-source-code traceability links using latent semantic indexing. In: Proceedings of the 25th international conference on software engineering, pp 125–135

Marcus A, Sergeyev A, Rajlich V, Maletic JI (2004) An information retrieval approach to concept location in source code. In: Proceedings of the 11th working conference on reverse engineering, pp 214–223

Marcus A, Maletic J, Sergeyev A (2005*) Recovery of traceability links between software documentation and source code. Int J Softw Eng Knowl Eng 15(5):811–836

Maron M, Kuhns J (1960) On relevance, probabilistic indexing and information retrieval. J ACM 7(3):216–244

McMillan C, Poshyvanyk D, Revelle M (2009*) 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, pp 41–48

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

Natt och Dag J, Gervasi V, Brinkkemper S, Regnell B (2004*) 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, pp 283–294

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

Oliveto R (2008) Traceability management meets information retrieval methods: strengths and limitations. PhD thesis, University of Salerno

Oliveto R, Gethers M, Poshyvanyk D, De Lucia A (2010*) On the equivalence of information retrieval methods for automated traceability link recovery. In: Proceedings of the 18th international conference on program comprehension, pp 68–71

Olsson T (2002) Software information management in requirements and test documentation. Licentiate thesis, Lund University

Park S, Kim H, Ko Y, Seo J (2000*) Implementation of an efficient requirements analysis supporting system using similarity measure techniques. Inform Softw Technol 42(6):429–438

Parvathy AG, Vasudevan BG, Balakrishnan R (2008*) 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, pp 64–69

Petersen K, Wohlin C (2009) Context in industrial software engineering research. In: Proceedings of the 3rd international symposium on empirical software engineering and measurement, pp 401–404

Petersen K, Feldt R, Mujtaba S, Mattsson M (2008) Systematic mapping studies in software engineering. In: Proceedings of the 12th international conference on evaluation and assessment in software engineering, pp 71–80

Pohl K, Bockle G, van der Linden F (2005) Software product line engineering: foundations, principles, and techniques. Birkhäuser

Ponte J, Croft B (1998) A language modeling approach to information retrieval. In: Proceedings of the 21st annual international SIGIR conference on research and development in information retrieval, pp 275–281

Port D, Nikora A, Hihn J, Huang L (2011*) Experiences with text mining large collections of unstructured systems development artifacts at JPL. In: Proceedings of the 33rd international conference on software engineering, pp 701–710

Randolph J (2005) Free-Marginal multirater kappa (multirater k[free]): an alternative to fleiss' Fixed-Marginal multirater kappa. In: Joensuu learning and instruction symposium

Robertson S (1977) The probability ranking principle in IR. J Doc 33(4):294–304

Robertson S, Robertson J (1999) Mastering the requirements process. Addison-Wesley Professional

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

Robertson SE, Jones S (1976) Relevance weighting of search terms. J Am Soc Inform Sci27(3):129–146

Rocchio J (1971) Relevance feedback in information retrieval. In: Salton G (ed) The SMART retrieval system: experiments in automatic document processing. Prentice-Hall, pp 313–323

Runeson P, Alexandersson M, Nyholm O (2007) Detection of duplicate defect reports using natural language processing. In: Proceedings of the 29th international conference on software engineering, pp 499–510

Runeson P, Höst M, Rainer A, Regnell B (2012) Case study research in software engineering. Guidelines and examples. Wiley

Sabaliauskaite G, Loconsole A, Engström E, Unterkalmsteiner M, Regnell B, Runeson P, Gorschek T, Feldt R (2010) Challenges in aligning requirements engineering and verification in a Large-Scale industrial context. In: requirements engineering: foundation for software quality, pp 128–142

Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Inf Process Manage 24(5):513–523

Salton G, Wong A, Yang C (1975) A vector space model for automatic indexing. Commun ACM 18(11):613–620

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

Settimi R, Cleland-Huang J, Ben Khadra O, Mody J, Lukasik W, DePalma C (2004*) Supporting software evolution through dynamically retrieving traces to UML artifacts. In: Proceedings of the 7th international workhop on principles of software evolution, pp 49–54

Shull F, Carver J, Vegas S, Juristo N (2008) The role of replications in empirical software engineering. Empir Software Eng 13(2):211–218

Singhal A (2001) Modern information retrieval: a brief overview. Data Eng Bull 24(2):1–9

Smeaton A, Harman D (1997) The TREC experiments and their impact on europe. J Inf Sci 23(2):169–174

Spanoudakis G, d'Avila-Garcez A, Zisman A (2003) Revising rules to capture requirements traceability relations: A machine learning approach. In: Proceedings of the 15th international conference in software engineering and knowledge engineering

Spanoudakis G, Zisman A, Perez-Minana E, Krause P (2004) Rule-based generation of requirements traceability relations. J Syst Softw 72(2):105–127

Spärck Jones K, Walker S, Robertson SE (2000) A probabilistic model of information retrieval: development and comparative experiments. Inf Process Manage 36(6):779–808

Stone A, Sawyer P (2006) Using pre-requirements tracing to investigate requirements based on tacit knowledge. In: Proceedings of the 1st international conference on software and data technologies, pp 139–144

Sultanov H, Huffman Hayes J (2010*) Application of swarm techniques to requirements engineering: Requirements tracing. In: Proceedings of the 18th international requirements engineering conference, pp 211–220

Sundaram S, Huffman Hayes J, Dekhtyar A (2005*) Baselines in requirements tracing. In: Proceedings of the workshop on predictor models in software engineering, pp 1–6

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

Torchiano M, Ricca F (2010) Impact analysis by means of unstructured knowledge in the context of bug repositories. In: Proceedings of the 4th international symposium on empirical software engineering and measurement, pp 47:1–47:4

Turtle H, Croft B (1991) Evaluation of an inference network-based retrieval model. Trans Inf Syst 9(3):187–222

Van Rompaey B, Demeyer S (2009) Establishing traceability links between unit test cases and units under test. In: Proceedings of the 13th European conference on software maintenance and reengineering, pp 209–218

Voorhees E (2005) TREC: Experiment and evaluation in information retrieval. MIT Press

Wang X, Lai G, Liu C (2009*) Recovering relationships between documentation and source code based on the characteristics of software engineering. Electron Notes Theor Comput Sci 243:121–137

Winkler S (2009*) Trace retrieval for evolving artifacts. In: Proceedings of the international workshop on traceability in emerging forms of software engineering, pp 49–56

Winkler S, Pilgrim J (2010) A survey of traceability in requirements engineering and model-driven development. Softw Syst Model 9(4):529–565

Wohlin C, Runeson P, M Höst, Ohlsson M, Regnell B, Wesslén A (2012) Experimentation in software engineering: a practical guide. Springer

Yadla S, Huffman Hayes J, Dekhtyar A (2005*) Tracing requirements to defect reports: an application of information retrieval techniques. Innov Syst Softw Eng 1:116–124

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

Zhai C (2008) Statistical language models for information retrieval a critical review. Foundations and Trends Information Retrieval 2(3):137–213

Zhai C, Lafferty J (2001) Model-based feedback in the language modeling approach to information retrieval. In: Proceedings of the 10th international conference on information and knowledge management, pp 403–410

Zhao W, Zhang L, Liu Y, Luo J, Sun JS (2003*) Understanding how the requirements are implemented in source code. In: Proceedings of the 10th Asia-Pacific software engineering conference, pp 68–77

Zhou X, Yu H (2007*) A clustering-based approach for tracing object-oriented design to requirement. In: Proceedings of the 10th international conference on fundamental approaches to software engineering, pp 412–422

Zou X, Settimi R, Cleland-Huang J (2006*) Phrasing in dynamic requirements trace retrieval. In: Proceedings of the 30th international computer software and applications conference, pp 265–272

Zou X, Settimi R, Cleland-Huang J (2008*) Evaluating the use of project glossaries in automated trace retrieval. In: Proceedings of the international conference on software engineering research and practice, pp 157–163

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

Markus Borg is a PhD student at Lund University and a member of the Software Engineering Research Group. His research interests are related to alleviating information overload in large-scale software development, with a focus on information retrieval and recommendation systems. Prior to his PhD studies, he worked three years as a development engineer at ABB in safety-critical software engineering. He is a student member of IEEE, and the Swedish research school in Verification and Validation (SWELL).

Per Runeson is a professor of software engineering at Lund University, Sweden, and is the leader of its Software Engineering Research Group (SERG) and its Industrial Excellence Center on Embedded Applications Software Engineering (EASE). His research interests include software development methods, in particular for verification and validation. He has co-authored handbooks for experiments and case studies in software engineering and serves on the editorial board of the Empirical Software Engineering Journal.

Anders Ardö is an associate professor and docent at department of Electrical and Information Technology, Lund University, where he is managing KnowLib—Knowledge Discovery and Digital Library Research Group. He founded the NetLab company in 1992 which marked itself as one of the internationally leading development laboratories within the emerging field 'digital libraries'. Current research interests include Information Retrieval and Knowledge Discovery.