International Journal of Software Engineering
and Knowledge Engineering
Vol. 22, No. 3 (2012) 1−49

DOI: 10.1142/S0218194012005846

World Scientific
www.worldscientific.com

REQUIREMENTS TRACEABILITY: A SYSTEMATIC REVIEW AND INDUSTRY CASE STUDY

RICHARD TORKAR*, TONY GORSCHEK, ROBERT FELDT,
MIKAEL SVAHNBERG, UZAIR AKBAR RAJA and KASHIF KAMRAN

Blekinge Institute of Technology, S-371 79 Karlskrona, Sweden
**richard.torkar@bth.se*
http://www.bth.be/com/serl

Received 18 February 2010
Revised 7 December 2010

Requirements traceability enables software engineers to trace a requirement from its emergence to its fulfillment. In this paper we examine requirements traceability definitions, challenges, tools and techniques, by the use of a systematic review performing an exhaustive search through the years 1997−2007. We present a number of common definitions, challenges, available tools and techniques (presenting empirical evidence when found), while complementing the results and analysis with a static validation in industry through a series of interviews.

Keywords: Requirements traceability; systematic review; case study.

1. Introduction

According to [75] requirements engineering involves activities for discovering, documenting and maintaining a set of requirements for a system. Requirements engineering (RE) activities are often divided into five categories: Elicitation, analysis, specification, validation and management.

Requirements management assists in maintaining a requirement's evolution throughout a development project. According to,[75] requirements management is concerned with all processes involved in changing system requirements and [34, 36] conclude that documentation, change management and traceability are the key activities of requirements management (RM), which is part of RE. One of the main tasks of RM is to assure requirements traceability (RT) from start throughout the artifact's lifetime. Traceability is also recommended as a necessary activity by e.g. IEEE Std. 830−1998 [49] and CMMI [23].

In [38] the authors define requirements traceability (RT) as "the ability to describe and follow the life of a requirement in both forward and backward direction

(i.e. from its origins, through its development and specification to its, subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases)."

The definition of RT by [38] is considered to be a comprehensive definition and is cited by several researchers and the same definition is also quoted at the Software Engineering Institute's website [71].

According to [37] there are two aspects of RT: Pre- and post-RS traceability. Pre-RS traceability is concerned with those aspects of a requirement's life from the point *before* it is included in the software requirements specification (SRS) [6]. In pre-RS traceability, requirements are related to their origin and other requirements. The origin of requirements includes stakeholders, business rules or previous documents. Post-RS traceability, on the other hand, is concerned with those aspects of a requirement's life from the point *after* it is included in the SRS [6]. Post-RS traceability ensures that all requirements are fulfilled. In post-RS traceability, requirements are often related to test cases that ensure that software items satisfy the requirements.

In addition to the above two aspects of RT, others have classified traceability into the following four types [74]:

- Backward-from traceability: Links requirements to their sources which are in e.g. other documents.
- Forward-from traceability: Links requirements to the design and implementation components.
- Backward-to traceability: Links design and implementation components back to the requirements.
- Forward-to traceability: Links other documents to relevant requirements, e.g. operation manuals describing the system functionality.

Initially RT was mostly used in the development of safety critical systems only. Nowadays RT has been proven to be beneficial in most types of development:

- Requirements traceability helps to identify the source of the requirement whether issued by a person or document or group of persons [63].
- Requirements traceability helps in performing impact analysis [1], which traces to what other component(s) might be affected in response to change in a particular requirement.
- Requirements traceability helps in test case verification, e.g. which test cases verify a certain requirement [63].
- Requirements traceability assists in tracking the overall progress of the project, e.g. one can measure how many requirements are implemented, how many are still in the design phase and how many are completed.
- Requirements are many times interdependent on each other and on other artifacts [12]. Requirements traceability helps in tracing this relationship.

This paper aims to establish the state of the art of requirements traceability as well as to identify the main challenges reported by research and industry. In order to achieve this a systematic review was performed covering the years 1997−2007. As a second step two case studies were performed to validate and complement the review findings.

1.1. *Research questions*

Two research questions will be answered through a series of questions connected to the review and its review protocol:

RQ 1. What is requirements traceability based on the state of the art in research and standards?

The research on requirements traceability will be carried out mainly by the use of a systematic review [55]. The focus of this systematic review will be on requirements traceability from 1997−2007. This will help in identifying, clarifying and understanding requirements traceability in general and requirements traceability techniques in particular.

RQ 2. What are the main factors reported by academia and industry as hindering (proper) implementation of requirements traceability?

Some of the factors, for instance lack of coordination between people, failure to follow standards or requirements traceability costs, may obstruct the implementation of requirements traceability policies; however, contributions from researchers will hopefully clarify to what extent, and why, this happens. Interviews will be conducted in industry to find out if there are any impeding factors which are not reported in the literature.

1.2. *Research methodology*

A mixed approach using both quantitative and qualitative research methodologies was adopted for this study. Through the systematic review, very much a quantitative method, requirements traceability is mapped as an area through a number of definitions and a number of issues regarding requirements traceability are presented. Even though the quantitative results were significant, there was still a need to complement the results with accompanying semi-structured interviews [70] carried out in industry in order to receive an early initial vaidation.

2. Systematic Review

A systematic literature review provides a mechanism for evaluating, identifying and interpreting all available research relevant to a particular research question, topic area or phenomenon of interest [55]. Individual studies contributing to the systematic review are known as primary studies while a systematic review is a secondary study.

A systematic review compromises three main phases: Planning the review, executing the review and reporting the review. In the planning phase the need for the systematic literature review is identified and a review protocol is developed. This

review protocol serves as a comprehensive search guide throughout the systematic literature review. Executing the literature review involves identification of research, primary studies selection, study quality assessment, data extraction and monitoring, and data synthesis. The phase of reporting the systematic literature review is a single stage phase.

The stages involved in a systematic literature review seem to be sequential but actually they may involve a number of iterations [55]. For example, many activities in the review protocol like search terms, inclusion and exclusion criteria for primary studies, are revised while the review is in progress [50].

The objective of this systematic review is to summarize the work done in RT during the years 1997−2007. The period was chosen in order to provide a good sample of the current knowledge in requirements traceability, it encompassed 3,087 articles in total as will later be seen, and it is not uncommon that a systematic review is delimited in this way (see e.g. [2, 29, 32, 43] for some recent examples).

2.1. *Development of review protocol*

The purpose of the study described in this review protocol is to review the current status of research in requirements traceability from 01 Jan. 1997 to 30 Sept. 2007.

Systematic review questions. The following questions (directly or indirectly connected to Research Questions 1 and 2) will be answered during the systematic review:

(1) What is requirements traceability based on state-of-the-art research?
(2) What are the challenges when implementing requirements traceability and how does research address these challenges?
(3) Which are the various requirements traceability tools according to research literature?
(4) What requirements traceability techniques are covered in research literature?

Search strategy. The search process was executed through online search using search terms and resources to be searched. The following search terms were used to extract primary studies:

(1) Requirements Traceability
(2) Requirements Traceability Technique
(3) Traceability
(4) Requirements Tracing
(5) 1 AND Challenges
(6) 1 AND Problems
(7) 1 AND Issues
(8) 1 AND Success Stories
(9) 2 AND Challenges
(10) 2 AND Problems

(11) 2 AND Issues
(12) 2 AND Success Stories
(13) 1 AND Experience Reports
(14) 1 AND Process Improvement
(15) 1 AND Impact
(16) 1 AND Experiences
(17) 1 AND Lessons Learned
(18) 1 AND Good Practices
(19) 1 AND Standards
(20) 1 AND Return on Investment
(21) 1 AND ROI

The following online resources were used during the systematic review:

- IEEE Xplorer
- ACM Digital Library
- Springer Link
- Inspec
- Compendex

In addition to the online search a number of conference proceedings and journals were manually investigated to reduce the risk of excluding an important article by mistake. The top-6 journals and top-3 proceedings showing up in the pilot searches are to be found in the first column in Table 1 and will be further elaborated later on in this section.

Study selection criteria and procedures

- Study Inclusion Criteria. The articles on RT published between 01 Jan. 1997 and 30 Sept. 2007 were included. The following questions were used to help judge the suitability:

(1) Is the article available in full-text?
(2) Is it a peer-reviewed article?
(3) Does it contain a case study, experiment, survey, experience report, comparative evaluation and/or action research?
(4) Does it report success, issues and/or failures or any type of experience concerning RT?
(5) Is it based on research done in the RT area?
(6) Does it contain definitions on RT?
(7) Does it introduce new and important claims regarding RT and supporting these claims with some sort of evidence?
(8) Does it identify problems and/or challenges in RT?
(9) Does it provide some sort of solution, roadmap or framework related to traceability problems?

Table 1. Articles on requirements traceability in selected journals and proceedings (the rightmost column). The second column lists the total number of articles found during the time span (1997−2007).

	Articles	Articles (RT)
Journal		
IEEE Transactions on Software Engineering (TSE)	751	4
ACM Transactions on Software Engineering Methodology (TOSEM)	139	1
Springer Requirements Engineering Journal	144	2
Springer Innovations in Systems and Software Engineering	48	2
Springer Software and Systems Modeling	106	1
Springer Annals of Software Engineering	150	2
Conference Proceeding		
ACM International Conference on Software Engineering (ICSE)	1,272	3
IEEE International Conference on Requirements Engineering (RE)	357	12
IEEE International Symposium on Requirements Engineering	120	2
Total	3,087	29

(10) Does it evaluate or compare two or more RT techniques?

- Study Exclusion Criteria.
  - — If the answer is "no" to question 1 or 2, the study is excluded immediately.
  - — At least questions 3–10 must be answered "yes".

Study selection process. The study selection process was based on the title, abstract and conclusion of the research paper. If it satisfied the inclusion criteria the complete research paper was read.

Study quality assessment and procedures. The selected articles were then evaluated based on the following criteria:

(1) Introduction. Does the introduction provide an overview of RT?
(2) Method. Was the research methodology clearly defined in the research paper?
(3) Results. Were the study results presented in a clear manner? Do the results help to solve an RT problem? What types of validity threats were defined for the results?
(4) Analysis. How was data analyzed? What type of analysis techniques were used? If the article contained a framework, then has it been validated in an industrial setting?
(5) Discussion and/or conclusion. Were negative findings properly reported? Were there any limits or restrictions imposed on the conclusions claims?

Data extraction strategy. In order to obtain the information from each primary study the following data extraction form was used:

(1) General information regarding research paper
   (a) Article title

(b) Author(s) name(s)
(c) Journal, conference proceeding
(d) Search terms used to retrieve research article
(e) Retrieval database of research article
(f) Date of publication

(2) Specific information regarding research paper

(a) Study environment

(i) Industry
(ii) Academia

(b) Research methodology

(i) Action research
(ii) Experiment
(iii) Case study
(iv) Survey

(c) Subjects

(i) Professionals
(ii) Students
(iii) Number of subjects
(iv) Subject selection

(d) Requirements traceability

(i) Definition of RT
(ii) Challenges with RT
(iii) Problems with RT
(iv) Solutions to RT problems

(A) Model or framework for RT
(B) Requirements traceability tools
(C) Number of RT techniques used in model and/or framework
(D) Name(s) of RT technique(s) used in model and/or framework
(E) Evidence regarding validation of proposed model and/or framework

(v) Requirements traceability techniques

(A) Evaluation of RT techniques
(B) Comparison of RT techniques

(e) Validity threats

(i) Conclusion
(ii) Construct
(iii) Internal
(iv) External

Synthesis of extracted data. In a systematic review, data synthesis is done by collecting and summarizing the results of the included primary studies. The studies included in a systematic review are different from each other based on their methodology and outcomes. These types of studies are known as heterogeneous. Due to the heterogeneous nature of the data a qualitative synthesis was used. The qualitative synthesis was done by reading and analyzing the research articles.

2.2. *Evaluating the review protocol*

The review protocol is a critical element of the systematic review and therefore an agreed validation process should be carried out in order to evaluate the protocol. [55] recommends conducting one or more pilot searches to identify the potential primary studies using search terms and search resources which are defined in the review protocol. The review protocol was peer-reviewed by three senior researchers which all had previous experience in conducting systematic literature reviews.

2.3. *Execution of review*

The selection of primary studies is a two-stage process. In the first stage the title, abstract and conclusion of a paper is studied and irrelevant papers are rejected. In the second stage the selected research papers are reviewed based on the inclusion and exclusion criteria defined in the review protocol to obtain a final list of primary studies. In this systematic literature more than 3,087 articles were scanned and initially 75 articles were selected. Then, after applying inclusion and exclusion criteria 52 articles were finally selected for a complete review (Table 2). The rejected articles are listed in [79].

Table 2. Primary studies as found by the systematic review.

Ref.	Title
[3]	A Feature-Oriented Requirements Tracing Method: A Study of Cost-Benefit Analysis
[4]	Recovering Traceability Links Between Code and Documentation
[5]	Overcoming the Traceability Benefit Problem
[8]	Traceability Techniques: A Critical Study
[9]	Deciding to Adopt Requirements Traceability in Practice
[10]	Requirements Traceability in Automated Test Generation Application to Smart Card Software Validation
[11]	Tracing Cross-Cutting Requirements via Context-Based Constraints
[17]	Supporting Event-Based Traceability through High-Level Recognition of Change Events
[18]	Automating Speculative Queries through Event-Based Requirements Traceability
[16]	Event-Based Traceability for Managing Evolutionary Change
[19]	Automating Performance-Related Impact Analysis Through Event-Based Traceability
[22]	A Heterogeneous Solution for Improving the Return on Investment of Requirements Traceability
[14]	Toward Improved Traceability of Non-Functional Requirements
[20]	Goal-Centric Traceability for Managing Non-Functional Requirements
[21]	Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability
[15]	Best Practices for Automated Traceability

Table 2. (*Continued*)

Ref.	Title
[57]	Recovering Traceability Links in Software Artifact Management Systems using Information Retrieval Methods
[25]	Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods
[27]	Design Traceability
[31]	Automating Requirements Traceability: Beyond the Record and Replay Paradigm
[38]	Extended Requirements Traceability: Results of an Industrial Case Study
[39]	Crafting the Requirements Record with the Informed use of Media
[40]	From Non-Functional Requirements to Design Through Patterns
[42]	TRAM: A Tool for Requirements and Architecture Management
[44]	Improving Requirements Tracing via Information Retrieval
[45]	Improving After-the-Fact Tracing and Mapping: Supporting Software Quality Predictions
[46]	REquirements TRacing On Target (RETRO): Improving Software Maintenance Through Traceability Recovery
[47]	A Case Study on Value-Based Requirements Tracing
[51]	Requirements Tracing
[52]	XTraQue: Traceability for Product Line Systems
[53]	Modeling Functional Requirements to Support Traceability Analysis
[54]	A Reusable Traceability Framework Using Patterns
[56]	Reconstructing Requirements Coverage Views from Design and Test Using Traceability Recovery via LSI
[58]	Recovery of Traceability Links Between Software Documentation and Source Code
[59]	Using Scenarios to Support Traceability
[60]	Enhancing Traceability Using Ontologies
[61]	Tool Support for Computer-Aided Requirements Traceability in Architectural Design: The Case of DesignTrack
[62]	Towards Method-Driven Trace Capture
[28]	Experiences Using Scenarios to Enhance Traceability
[63]	Factors Influencing Requirements Traceability Practice
[65]	Requirements Traceability: Theory and Practice
[64]	Toward Reference Models for Requirements Traceability
[66]	Pre-Requirement Specification Traceability: Bridging the Complexity Gap Through Capabilities
[68]	An Evaluation of Traceability Approaches to Support Software Evolution
[69]	Improving Software Quality Through Requirements Traceability Models
[72]	A Framework for Managing Traceability Relationships Between Requirements and Architectures
[73]	Scenario Advisor Tool for Requirements Engineering
[76]	Rule-Based Generation of Requirements Traceability Relations
[77]	Traceability for System Families
[80]	Introducing Efficient Requirements Management
[81]	Requirements Traceability to Support Evolution of Access Control
[83]	Tracing Requirements to Defect Reports: An Application of Information Retrieval Techniques

The search was carried out in two steps. During the first step the selected electronic resources were explored online using the search terms defined in the review protocol. Research papers related to RT were downloaded and their details were recorded. In the second step the selected conference proceedings and journals were searched manually year by year (Table 1). Research papers related to RT were first searched in the records. If they were not found in the records they were then downloaded.

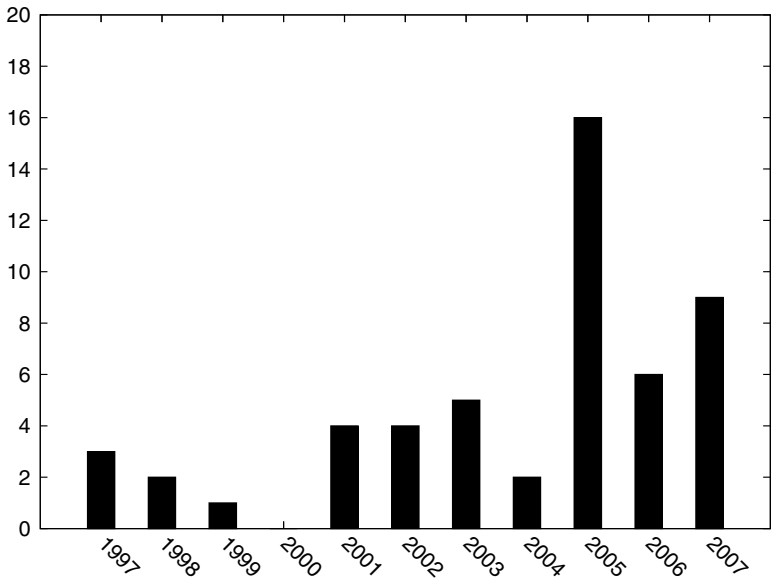

Fig. 1. Year-wise distribution of primary studies on requirements traceability during 1997−2007.

Only three research papers [46, 52, 73] were found by manually exploring the sources listed in Table 1. These papers were downloaded and their details recorded. Concerning the distribution of research papers in relation to our selected journals and proceedings please see Table 1 (obviously, all papers in the rightmost column are included in Table 2).

Figure 1 represents the yearly distribution of the primary studies during the period 1997−2007. It is evident from the figure that the number of publications on RT peaked in the year 2005. One of the reasons for this increase in research publications (which can already be seen before 2005) is the International Workshop on Traceability in Emerging Forms of Software Engineering held in 2002, 2003, 2005 and 2007 (TEFSE'07 is not part of this search since the publications had not been published electronically when the study was executed). At this workshop only research papers related to traceability were presented and later published electronically.

3. Results from and Analysis of the Systematic Review

3.1. *Definitions of requirements traceability* (*Question 1*)

In [38] RT is defined as, "the ability to describe and follow the life of a requirement, in both a forward and backward direction (i.e. from its origin, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases)." This definition is further used by, e.g. [8, 10, 11, 27, 31, 47, 51, 28, 64−66, 68, 81]. The definition seems to be

more precise and comprehensive compared to other definitions reported in literature. It describes both pre- and post-RS traceability.

The IEEE Std. 830−1998 defines traceability as, "a software requirements specification is traceable if (i) the origin of each of its requirements is clear and if (ii) it facilitates the referencing of each requirement in future development or enhancement documentation" [49]. This definition covers both pre-RS traceability and post-RS traceability.

The U.S. Department of Defense Standard 2167A [26] defines traceability as, "that the document in question is in agreement with a predecessor document to which it has a hierarchical relationship." Additionally, traceability has five elements, according to DOD-STD-2167A:

- The document in question contains or implements all applicable stipulations of the predecessor document,
- A given term, acronym, or abbreviation means the same thing in all documents,
- A given item or concept is referred to by the same name or description in the documents,
- All material in the successor document has its basis in the predecessor document, that is, no untraceable material has been introduced, and
- The two documents do not contradict one another.

This definition provides a solid base for researchers to understand and interpret RT; however, the definition does not make a clear distinction between pre- and post-RS traceability.

Edwards and Howell define RT as "a technique used to provide a relationship between the requirements, the design and the final implementation of the system." This definition reflects the traces from requirement to other artifacts, which means that it focuses on post-RS traceability, but on the other hand ignores pre-RS traceability [30].

In [7], the authors define traceability as providing an "essential assistance in understanding the relationship that exists within and across software requirements, design and implementation". This definition is similar to the previous definition by Edwards and Howell, i.e. it focuses only on post-RS traceability.

Hamilton and Beeby define traceability as, "the ability to discover the history of every feature of a system so that the changes in the requirements can be identified". This definition covers both pre- and post-RS traceability [41].

In [64] and [76] the authors once again focus on post-RS traceability only. The latter, define traceability as "the ability to relate requirements specifications with other artifact created in the development life-cycle of a software system". The former, define traceability as a "property of a system description technique that allows changes in one of the three system descriptions — requirements, specifications, and implementation — to be traced to the corresponding portions of the other descriptions".

Table 3. Primary studies defining or using requirements traceability definitions.

Ref.	Definition used
[5]	Arkley & Riddle
[8]	Spanoudakis, Murray, Ramesh, Gotel & Finkelstein and IEEE Std. 830−1998
[10]	Gotel & Finkelstein
[11]	Gotel & Finkelstein
[17]	Gotel & Finkelstein
[19]	IEEE Std. 830−1998
[20]	Gotel & Finkelstein
[15]	Gotel & Finkelstein
[27]	Gotel & Finkelstein
[31]	Gotel & Finkelstein
[38]	Gotel & Finkelstein
[47]	Gotel & Finkelstein
[51]	Gotel & Finkelstein
[53]	DOD-STD-2167A and IEEE Std. 830−1998
[28]	Gotel & Finkelstein
[65]	IEEE Std. 830−1998 and Gotel & Finkelstein
[64]	Gotel & Finkelstein, Edward & Howell and Hamilton & Beeby
[66]	Gotel & Finkelstein
[68]	Gotel & Finkelstein
[76]	Ramesh & Jarke
[81]	Gotel & Finkelstein

3.1.1. *Analysis of definitions*

The primary studies containing definitions on RT gathered by the systematic review are presented in Table 3. In order to perform an analysis of RT definitions we add two parameters 'scope of definition' as a column in Table 4. The column "scope of definition" provides the aspect of traceability i.e. pre-RS traceability and/or post-RS traceability.

Hence, the conclusion drawn from Table 4 is that RT is fully defined when both aspects of pre- and post-RS traceability are covered, and that the definitions from

Table 4. Scope of different definitions as found in the primary studies.

Definition	Scope of definition
Gotel & Finkelstein	Pre- and post-RS traceability
IEEE Std. 830−1998	Pre- and post-RS traceability
DOD-STD-2167A	Pre- and post-RS traceability
Hamilton & Beeby	Pre- and post-RS traceability
Ramesh & Jarke	Post-RS Traceability
Edward & Howell	Post-RS traceability
Spanoudakis	Post-RS Traceability
Murray	Post-RS Traceability
Ramesh	Post-RS Traceability

Gotel and Finkelstein, Hamilton and Beeby, DOD-STD-2167A and IEEE Std. 830−1998 accomplish this.

It is evident from Table 3 that Gotel and Finkelstein's definition of RT is dominant (more than 80% of the primary studies, which defined requirements traceability, used this definition).

3.2. *Challenges in requirements traceability* (*Question 2*)

In this subsection challenges in requirements traceability will be covered (as found in the primary studies). Any techniques or tools discussed will be covered in subsequent subsections.

In [16], the authors discuss the results of a survey conducted by Gotel and Finkelstein in 1994 [37]. Various traceability problems were identified in this survey, such as informal development methods, insufficient resources, time and cost for traceability, lack of coordination between people which were responsible for different traceable artifacts, lack of training in RT practices, imbalance between benefits obtained and efforts spent for implementing traceability practices, and failure to follow standards. Cleland-Huang *et al.* comment that all of these issues are intensified by the challenges of today's distributed development environment. In order to solve some of these challenges they propose a traceability technique called event-based traceability (EBT). This technique creates links between software artifacts after a change request is executed, and, according to the authors, alleviates the coordination efforts required for maintaining software artifacts. Cleland-Huang *et al.* also recommends other techniques, e.g. information retrieval (IR), when working with automated RT.

In [39], the authors describe traceability problems such as, requirements changed by users, and availability of less contextual information in decision making. Gotel and Morris suggests a new dimension in the field of RT and media to solve these problems. They propose a theoretical framework which helps to select the appropriate media for recording requirements-related information.

In [5], the authors claim that various project managers and team members perceive that RT does not offer immediate benefit to the development process; therefore RT is kept at low priority. They propose a new technique, traceable development contract (TDC), to overcome traceability problems. (TDC is used to control the interaction of development teams and to document traceability relationships.)

In [21], the authors claim that manual construction and maintenance of a traceability matrix proves to be costly for various reasons and, hence, it is a common perception that traceability is not feasible from a financial point of view. In order to solve this problem, dynamic retrieval methods are used to automate the generation of traceability links.

In [22], the authors present the problem of link maintenance, i.e. the lack of coordination between team members results in a failure in maintaining links between artifacts. Most of the time developers believe that traceability costs more than it

delivers. However, Cleland-Huang *et al.* point at the fact that excessive usage of traceability can also lead to confusion and they then present a combination of traceability techniques in a framework named traceability for complex systems (TraCS). They argue that TraCS helps to implement RT practices in a cost effective manner and bring significant value to an organization.

Reference [38] report the results of an empirical study related to the problems of RT. They identify a set of problematic questions such as, 'Who identifies or discovers a requirement and how? "Who was responsible for the requirement to start with, and who is currently responsible?" Who will take care of change(s) in requirements? What will be the effect on the project in terms of knowledge loss, if some individual or the whole group leave the company? Gotel and Finkelstein then propose a model using contribution structures to solve these issues. Contribution structures reflect the network of people who participate in producing artifacts in requirement engineering. This model extends the artifact-based traceability with traceability of people involved in requirements engineering. This model is then successfully implemented in a company as a case study [38]. The employees of the company commented that the proposed model identified the relevant people to rectify the specific problems regarding changes in requirements. The model has successfully been used in decision-making concerning staff turnover issues.

In [9], the authors investigate how practitioners (project managers) decide on when and how to adopt RT. They use a literature review combined with Howard's theory of classical decision making [48] to identify relevant factors for making a decision with respect to traceability. They then validate the factors in a large case study. The main conclusions they reach is that "There is little incentive to use traceability when most of the benefits are outside the project" and that there is a lack of awareness. In addition, they claim that one major obstacle is, "the way software development projects [are] contracted and organized."

In [80], the author presents experiences from a project related to requirements management improvement (specifically RT). Initially the problems related to requirements management were identified in the information transfer from contract to specification phase as well as inadequate impact analysis of requirement changes. In this regard three requirements management tools, i.e. DOORS, RTM and RequisitePro were evaluated and, ultimately, DOORS was recommended for solving traceability issues. (Requirements traceability tools are further covered in Sec. 3.3).

In [63], the author identifies environmental, organizational and technical factors influencing the implementation of RT. The environmental factors include inability to use available technologies for RT, such as reluctance to manually construct a requirements traceability matrix (RTM) by the employees. Organizational factors include compliance with standards strictly demanding RT like CMMI level 3. Technical factors include *ad hoc* practices and staff employed in organizations. Ramesh proposes tools support, training, and change in system development policies to overcome these challenges.

In [47], the authors identify costs associated with traceability as a factor that obstructs its implementation. The traceability costs significantly increases if a company uses RT tools. The reason for this increase in cost is that existing traceability techniques do not differentiate between high and low value requirements (the high value requirements are those that are more important as compared to other requirements and *vice versa*). The authors propose a method called value-based requirements tracing (VBRT) to solve this cost-related issue of RT. The VBRT approach identifies high value requirements based on parameters like: Number of artifacts, number of traces (links between software artifacts), and number of requirements. They successfully implement this approach in an industrial case study.

In [14, 20], the authors identify that organizations fail to trace non-functional requirements (NFRs) like performance, security and usability. This is, according to the authors, due to the fact that NFRs have a global impact on the software system and extensive network of interdependencies and trade-offs exist between them. In order to overcome these NFR-related traceability issues an approach called goal-centric traceability (GCT) is proposed. An industry-based experiment was conducted to verify this technique. The experimental results indicate that this approach is successful in managing traceability in NFR.

In [66], the authors identify one major problem when tracing requirements back to their sources. The system validation testing is performed against requirements; therefore, there should be a technique to trace requirements back to their sources. The authors propose a technique for pre-RS traceability.

3.2.1. *Analysis of challenges in requirements traceability*

Over time various researchers have reported different challenges regarding implementing requirements traceability practices. Among these challenges some can be referred to as commonplace, e.g. cost, time, effort, and team coordination. In order to tackle these challenges, various solutions have been proposed such as new traceability techniques, frameworks, models, and various automated traceability tools. The challenges related to RT and their solutions can be classified into three main types on the basis of the primary studies in the review:

- Challenges addressed by academia (Table 5).
- Challenges addressed by academia/industry and verified in industry (Table 6).
- Unsolved issues. Issues which are still unsolved in industry and academia.

The solutions proposed might not be generic, i.e. suitable for all types of companies, however, some general conclusions can be drawn. First, understanding the challenges that might arise when realizing and working with traceability can be seen as a pre-requisite. Second, the cost and value distribution of RT seems to be off, namely many of the benefits associated with RT span over and beyond any single project, while large parts of the costs associated with establishing and maintaining RT are put on the project. This could hamper the motivation of a project to work

Table 5. Primary studies reporting challenges related to requirements traceability addressed by academia.

Ref.	Issue(s)	Proposed solution (not verified in industry)
[5]	Requirement traceability does not offer immediate benefit to development process.	Traceable development contract.
[16]	Informal development methods, Insufficient resources, Time and cost for traceability, Lack of coordination between people, Failure to follow standards.	Event-based traceability.
[22]	Lack of coordination between team members. Developers think that traceability costs more then it delivers. Excessive use of traceability generate more links which are not easy to manage.	Traceability for Complex Systems (TraCS) Framework.
[21]	Manual construction of an RTM is costly.	Solved by the use of IR methods.
[39]	Requirements change by user. Less appropriate information is available for making decision with requirements.	Media recording framework.
[66]	Problems associated with tracing back to their sources.	Pre-RS requirements traceability technique.

Table 6. Primary studies reporting challenges related to requirements traceability addressed by industry.

Ref.	Issue(s)	Proposed solution (verified in industry)
[9]	Adopting RT	Increase awareness and adapt organizations to include RT.
[14]	Failure to trace non-functional requirements e.g. security, performance and usability.	Goal centric traceability evaluated by an experiment.
[38]	Some problematic questions are identified as challenges: Who identifies a requirement and how? Who was responsible for the requirement to start with and who is currently responsible? Who is responsible for change(s) in requirements? What will be the effect on the project in terms of knowledge loss if key employers are quit?	Framework of contribution structure.
[47]	Cost related to RT.	VBRT tested through a case study.
[63]	Organizational, environmental and technical factors.	Best practices given.
[80]	Requirements management challenges in industry projects e.g. inadequate impact analysis and lack of information transfer.	Requirements management tools like DOORS and RequisitePro.

with RT as the benefit is not seen immediately. This could be an underlying factor that enhances many of the other challenges identified.

3.3. *Requirements traceability tools* (*Question 3*)

This subsection will cover the requirements traceability tools found in the primary studies. Any traceability techniques discussed in this subsection will be covered in the following subsection.

RETRO. Requirements tracing on-target (RETRO) is an RT tool facilitating the automatic generation of requirements traceability matrices (RTM). RETRO uses information retrieval (IR) methods and has a GUI front-end [45, 46, 83].

Hayes *et al.* [46] conducted a case study with 30 graduate-level students in a requirements engineering course. The subjects in this case study were divided into two groups. One group was tracing requirements manually using a RTM, while the other group was using RETRO. Students who had previous experience of traceability were placed in the former group. Both groups had to trace 22 requirements to 52 design elements. The results of the case study revealed that the students using RETRO produced the most accurate result.

Rational RequisitePro. Rational RequisitePro is a requirements management tool developed by IBM. It provides support to save SRS documents and, link requirements to use-case diagrams and test cases. When change to requirements occur Rational RequisitePro identifies the corresponding software artifacts that are affected.

It is currently in use in industry and by researchers, e.g. [17, 18, 16, 22, 15, 44, 45] have all identified it as a requirements management tool, which also supports traceability.

DOORS. DOORS, is a requirements management tool developed by Telelogic (now IBM). It is used to capture, link, trace, analyze, and manage changes to the requirements. It provides ways to create and traverse links between requirements (for example a link can be created by a drag and drop operation). DOORS is also helpful in change management; it immediately flags the changes that could impact other requirements. In addition to all this DOORS also support dynamic report generation.

From 1997 to 2007 several researchers have reported using DOORS as an automated requirements management tool. DOORS has been covered implicitly or explicitly by several researchers [5, 15−18, 22, 44, 45, 64, 80].

DesignTrack. According to [61], DesignTrack is a prototype tool supporting RT. It provides traceability between requirements and architectural design. DesignTrack helps organizations by providing an integrated environment for requirements modeling and specification. An architect can use this tool to manage issues of design requirements and other design tasks.

The primary goals of DesignTrack is to: (i) Provide an integrated environment for designers to manage requirements information along with exploration. (ii) Facilitate

the designers to reuse previous requirement information. (iii) Permit designers to track changes as they go through requirements specifications or from explorative tasks.

TRAM. TRAM, or tool for requirements and architectural management, is a tool for managing system architecture, system requirements and the traceability between them. This tool is based on an information model identifying the relationship between requirements engineering and architectural design.

According to [42], TRAM has been used in case studies indicating positive results, however, these positive results are not explicitly discussed.

Scenario Advisor Tool. In software engineering, scenarios can be used to model system functionality. Scenarios act as mediators between requirements engineers and software artifacts like specifications and design documents. The scenario advisor tool provides traceability support between scenario models and requirements. It also facilitates in generating new scenarios and scenario variations [73].

In [73], an empirical study is reported where the authors try to determine the usefulness of the scenario advisor tool in scenario-based traceability. An experiment was conducted with two groups. One group used the scenario advisor tool, while another did not use tool support. The subjects of the group without tool support were eight postgraduate students and two researchers. In the group using the scenario advisor tool the subjects were nine postgraduate students and one researcher. The members of both groups had no previous experience in traceability using scenarios. The results of this experiment indicated that the scenario advisor tool helped users to write scenarios without any domain knowledge, i.e. users can write better scenarios and can trace requirements to scenario models. In the debriefing interviews some users reported that the tool could be improved if it provided a scenario template or a step-by-step procedure to write good scenarios.

Other traceability tools. The RT tools listed in Table 7 were not sufficiently described by the contributions found in the systematic review. Neither detailed information regarding how they worked nor any empirical evidence related to them was reported by the literature.

Table 7. Primary studies reporting on traceability tools lacking empirical evidence or sufficient description. The tool (second column) was introduced in the article(s) found in the first column.

References	Tool
[16, 64]	SLATE
[16]	CRADLE
[44, 45, 64]	RDD-100
[64]	Marconi RTM
[64]	RTS
[64]	Rtrace
[64]	Teamwork/RQT

3.3.1. *Analysis concerning requirements traceability tools*

Requirements traceability tools help in maintaining traceability by, e.g. automatically generating links between various software artifacts and requirements. DOORS and Rational RequisitePro are widely used requirements management tools which also provide traceability support.

Tools like DesignTrack and TRAM provide traceability between requirements and architecture. On the other hand Scenario Advisor Tool provides traceability support between requirements and scenarios and RETRO is a pure traceability tool facilitating automatic generation of RTM.

The results of the systematic review reveal that all of the above traceability tools except DesignTrack have been evaluated empirically. Furthermore, it seems that requirements management tools are widely used instead of tools focusing exclusively on traceability support. Table 8 provides a comparison of the RT tools found in the systematic review (containing a sufficient description or providing empirical evidence).

Based on the results of the systematic review, the RT tools can be classified into three categories:

First, requirements management tools providing traceability support. This category includes tools that are developed to facilitate requirements management

Table 8. Comparison of found requirements traceability tools sufficiently described or providing empirical evidence.

Refs.	Tool	Description	Type	Empirical evidence
[5, 15−17, 22] [25, 44, 64, 80]	DOORS	It is used to capture, link, trace, and manage changes to requirements.	Requirements management tool that supports traceability.	Currently used in industry.
[15−18] [22, 25, 44]	Rational RequisitePro	Provides support to link requirements to use-case diagrams, test cases and save SRS in its database.	Requirements management tool providing traceability support.	Currently used in industry.
[42]	TRAM	Managing system architecture and system requirements.	It supports traceability between them.	Case studies
[45, 46, 83]	RETRO	Facilitate automatic generation of RTM.	Traceability tool.	Case study [46].
[61]	DesignTrack	Traceability between requirements and architectural design.	Requirements management tool supporting RT.	No Empirical Evidence, however sufficiently described.
[73]	Scenario Advisor Tool	Traceability between scenario models and requirements.	Helps in writing scenarios but supports traceability.	Experiment.

activities in general. Traceability is a part of the requirements management process and, hence, these tools also support traceability. This category includes tools like Rational RequisitePro, DOORS, DesignTrack and the Scenario Advisor Tool.

Second, requirements traceability tools. This category includes tools that are used exclusively for managing RT, e.g. tools like RETRO and TRAM.

Third, other requirements management and traceability tools. This category includes those tools that are only reported casually by the systematic review articles, i.e. the articles gathered by the systematic review do not describe details regarding the tool nor empirical evidence regarding their use. This category includes tools as listed in Table 7. It is worth noticing that eight out of a total of 13 tools covered in the review were lacking any actual validation, either in industry or in a lab environment [35]. We are not able to positively draw the conclusion that no validation exists as it could be covered publication not included in this review. Although the fact that more than 60% of the tools found have no validation in the associated publications might be an indication of inadequate validation, in which case the usability and usefulness of the tools have not be tested.

3.4. *Requirements traceability techniques*

This subsection covers the requirements traceability techniques found in the primary studies of the systematic review.

Value-based requirements tracing. Many existing tracing approaches make no distinction between requirements that are very valuable to trace and requirements that are less valuable to trace. This increases the efforts related to RT and therefore seems more costly to implement in practice [47]. The tracing value depends on several parameters like stakeholder importance, risk or volatility of requirement and the necessary tracing cost. The value-based requirements tracing (VBRT) technique takes these parameters into consideration.

Ramesh [63] identify two types of traceability users namely low-end and high-end traceability users. The low-end traceability users capture traceability information uniformly for all requirements, therefore, in many cases, they also treat traceability as expensive. On the other hand high-end traceability users recognize that all requirements are not equal with respect to their criticality or significance. Therefore, they maintain traceability for critical project requirements to keep cost under control but still achieve some traceability benefits.

The goal of VBRT is to identify traces or traceability links based on prioritized requirements. The identification of traces early in the project life-cycle is easier than in the later stages. VBRT reduces the traceability efforts for the prioritized requirements according to [47]. The VBRT process consists of five steps as described below [47] and illustrated in Fig. 2.

During the requirements definition the project manager or requirements engineer analyzes the software requirements specification to identify atomic (single

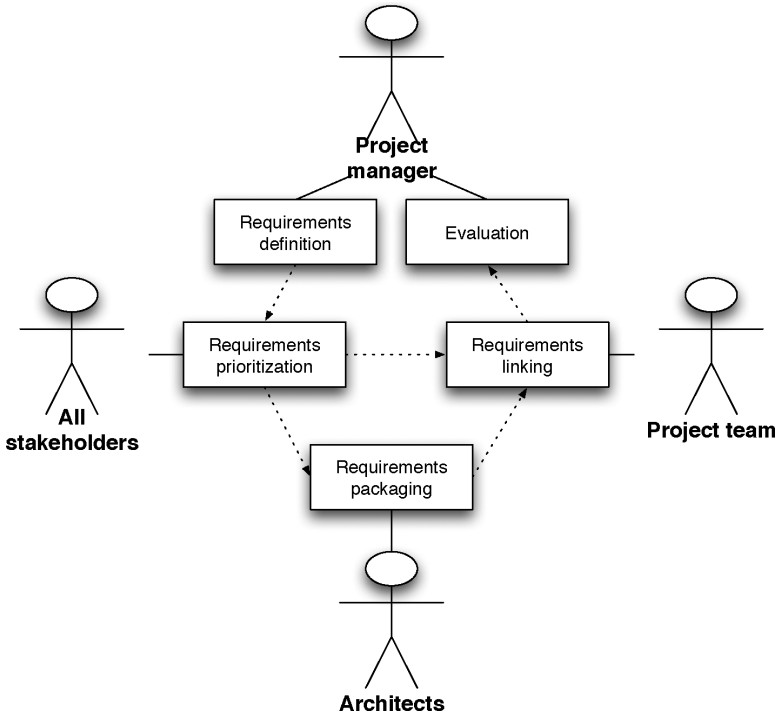

Fig. 2. Overview of the value-based requirements traceability process. The dashed arrows indicate an explicit or implicit connection between the activities. The actor Project manager can be substituted in some cases by, e.g. a requirements engineer.

irreducible) requirements. A unique identifier is assigned to each requirement by the requirements engineer. The result of the requirements definition step is a set of requirements and their IDs.

In the requirements prioritization phase all stakeholders assess the requirements based on three parameters, i.e. the value, the risk and the effort of each requirement. The result of this phase is an ordered list of prioritized requirements based on the three priority levels.

The packaging of requirements is optional and allows a group of architects to identify clusters of requirements. These clusters of requirements help to develop and refine an architecture from a given set of requirements.

During the linking of artifacts the team identifies traceability links between requirements and artifacts. Important requirements are traced in more detail than other requirements. These important requirements can be identified from the list of prioritized requirements based on three levels developed in the requirements prioritization step. A traceability plan is the by-product of this phase.

Finally, during the evaluation phase, a project manager can use traces for various purpose such as to estimate the impact of a change.

Heindl and Biffl [47] conducted a case study using VBRT at Siemens, Austria. The project Public Transport on Demand consisted of 46 requirements that were selected to compare full tracing and VBRT.

The results of the case study were:

(1) Focusing on the most important requirements reduced efforts as compared to tracing all requirement. The high effort spent in requirements tracing is the factor due to which projects do not implement traceability. The effort can be reduced using VBRT, which took 35% less effort compared to full tracing.
(2) It is easier to capture traceability information in earlier phases of the software development life-cycle as compared to capturing traceability in later stages. (For example if no traceability information is maintained during the project.)
(3) The prioritization step of VBRT identifies the most valuable requirements that are needed to be traced in more detail.

Feature-oriented requirements traceability. Artifacts, like software requirements specification, design documents, source code and test cases, are produced during software development. When a change request (CR) occurs at any stage during the software development life-cycle, it might be difficult to discover the software artifacts affected by the CR. Additionally, it may be difficult for software engineers to construct and manage traceability links in general. The approach of the feature-oriented requirements tracing (FORT) technique aims at reducing the difficulty in managing traceability links by identifying them through prioritized requirements and by constantly considering cost and efforts [3].

In order to understand the concept of FORT some terms need to be introduced as given below [3]:

- Features are the key characteristics of the product. These features can be classified on the basis of capabilities, domain technologies, implementation technologies and operating environments.
- The process of identifying features and then organizing them in a model called a feature model.
- The user visible characteristics that can be identified as operations, nonfunctional characteristics and distinct services.
- The domain technologies represent the way of implementing a service or an operation.
- Generic functions or techniques that are used to implement domain functions, services, and operations are called implementation techniques.
- The environments in which the applications are in use are called operating environments.
- There are three types of relationships in feature modeling. The composed-of relationship is used when there is a whole-part relationship between a feature and its sub-features. In generalization/specialization relationship features are the

generalizations of sub-features. Where as 'implemented-by relationship' is used when one feature is necessary to implement the other feature.

The FORT process then consists of five phases [3]:

First, requirements definition which in its turn consists of three activities: Analyzing requirements specification, identifying atomic requirements and assigning an identifier to each requirement. The aim of the requirements definition phase is to normalize user requirements to map them to various artifacts. In order to achieve this the requirements specification is analyzed and atomic requirements are identified. A unique identifier is then associated with each requirement. The result of this phase is a list of requirements with identifiers.

Next, feature modeling which consists of three activities: Identifying categories and features, organizing feature diagrams and assigning requirements related to features. According to [3], feature diagrams are developed by identifying the categories in the target system and features in each category. At this stage relationships between features are also taken into consideration. Finally all requirements are assigned to each feature. The result of this phase is a feature diagram and a list of features.

Third, is feature prioritization, which consists of two activities: Estimating values of requirements and ordering a list of features. In the feature prioritization phase stakeholders estimate the requirements based on value, risk and effort for each requirement. Next, the features are prioritized. (FORT provides a scale for feature prioritization as shown in the Table 9 [3].)

Fourth, there is the requirements linking phase. This phase consists of three activities assigning artifacts to related feature, breaking down implementation elements in different levels and establishing RT links. In requirements linking, RT links are generated and all artifacts are assigned to the related features. Then, implementation items are broken down by granularity levels and traceability links are established. These traceability links are actually the relationships among requirements, features and artifacts. Similarly important requirements are traced in more detail compared to less important requirements. This phase results in a list of traceability links.

Finally, we have the traceability links evaluation phase that consists of two activities: Actual usage of traceability links in during development and refinement of traceability links. In short, traceability links are used for conflict analysis, change impact analysis and consistency checking in during development. Based on this evaluation, the traceability links can be changed, removed or added.

Table 9. Priority levels and classification of artifacts.

Level	Granularity	Classification of artifacts
1	Low	Components
2	Medium	Class
3	High	Methods

Table 10. Results of case study on car rental system. No. of traceability links (TL) corresponds to $TL = l * m * n$, where l is the number of components, m is the number of classes and n is the number of methods. Efforts of generating traceability links (ETL) is $ETL = l[*m, [*n]]/TR * 100$.

Granularity	TL	ETL
Low	9	4
Medium	49	28
High	152	100

The authors in [3] provide empirical evidence related to FORT. Feature-oriented requirements tracing was applied in a case study to a car rental system containing nine components, 49 classes and 152 methods. The results of this case study are seen in Table 10.

The case study's result indicate that FORT reduces efforts for generating traceability links with 24−72% [3].

In short, FORT provides variability information by the use of feature modeling. This variability is helpful to estimate the impact of requirements change. FORT also reduces the efforts to create traceability links by prioritizing the features and, additionally, provides a tight relationship between requirements and artifacts by the help of an intermediate catalyst.

We have made a table between value-based and feature-oriented requirements tracing so as to clearly see the differences, Table 11 [3], since the two techniques are fairly similar to each other at first glance.

Pre-RS requirements traceability. In pre-RS tracing, requirements are traced back to their source. These sources are basically the user needs that, in most cases, corresponds to unstructured information. Requirements engineers use interviews, questionnaires or prototyping to gather user needs during the process of requirements elicitation. These needs are then documented as requirements in an SRS. Traceability between requirements and their sources is one of the biggest challenges

Table 11. Comparison between value-based and feature-oriented requirements traceability.

Criteria	VBRT	FORT
Well-defined process	Partial support	Supports
Value-added tracing	Supports	Supports
Feature modeling	No support	Supports
Variability consideration	No support	Supports
Efforts reduction	High	High
Tool support	Yes	No
Empirical evidence	Case study	Case Study

faced by the research community according to [66]. In system validation the system is validated against the requirements and in the case of a test failing it might be necessary to trace a requirement back to its sources.

The approach of pre-RS traceability, as explained in [66], is based on capabilities engineering. Capabilities engineering is a process for developing change-tolerant systems by using functional abstractions known as capabilities.

The capabilities engineering process is based on three phases:

First, the problem space that represents the conceptual region which is associated with the problem domain [66]. There are two important entities in the problem space: Needs and directives. The needs represent the user's view of the system. The needs specify what is desired of the system from user's perspective, expressed in the language of the problem domain. Directives are the detailed characteristics of the system or requirements with context information. There are two purposes of directives in the problem space. The first is to capture domain information and the second to facilitate progress from problem space to transition space. In the problem domain needs are decomposed into directives. Decomposition is achieved with the help of functional decomposition (FD) graphs and is a directed acyclic graph represented as $G = (V, E)$, where V is the vertex and E is the edge. The FD graph's root represents the overall mission of the system and edges represent the directives. The internal nodes between root and leaves are the functional abstractions. The FD graph provides traceability links between needs and directives.

Second, we have the transition space which, according to [66], is defined as a collective aggregation of the system view, capabilities and problem domain. The two main entities in the transition space are initial and optimized capabilities. Formulation and optimization are two capabilities engineering activities in transition space. The initial capabilities are the functional abstractions with high cohesion and low coupling whereas the optimized capabilities are the constraints of technology feasibility and implementation schedules. The formulation activity identifies initial capabilities from all possible abstractions present in the FD graph. These initial capabilities show high cohesion and low coupling. Cohesion represents the togetherness of elements within the entity and coupling is the interdependences between elements. The aim of the optimization activity is to identify the set which best accommodates the constraints of schedule and technology. The initial capabilities are the input for optimization activities.

Finally, there is the solution space that represents the technical area relevant to the system being developed. The important entity in the solution space is the requirement. In the solution space a requirement represents users' expectations from the system and is subject to quality constraints, e.g. testability, verifiability, accuracy, and un-ambiguity [66]. The input to the solution space is the optimized set of capabilities with their directives that are then transformed into requirements. In capabilities engineering based pre-RS traceability, only directives associated with the capabilities chosen for development are transformed into requirements. The relevance value can be used to determine critical requirements while transforming

Table 12. Pre-RS traceability using the capabilities engineering (CE) approach.

CE phase	Entities	CE activity	Input	View
Problem space	Needs, directives	Decomposition	User needs	User
Transition space	Initial and optimized capabilities	Formulation, optimization	FD graph	System
Solution space	Finalized capabilities (requirements)	Transformation	Optimized capabilities	System

directives to requirements and represents the importance of a directive in achieving the objective of its parent node. For instance, a directive with relevance 1 is mission critical and therefore the requirements associated with this directive are also mission critical.

Table 12 provides a summary of the pre-RS traceability process using the capabilities engineering approach. In [66] the authors do not provide any empirical evidence regarding the success of this approach, however they claim that this approach is very useful in RT.

Event-based traceability. The event-based traceability approach (EBT) was proposed by Cleland-Huang *et al.* in [18, 16]. The main reason for developing EBT was to provide maintenance of traceability relationships. Cleland-Huang *et al.* define traceability relationships as publisher-subscriber relationship. In this relationship, dependent objects, i.e. artifacts, have to subscribe to their respective requirements on which they are dependent. Whenever a requirement change occurs, an event message is published, which is then notified to all dependent objects.

According to [17], there are three main components which participate in the whole process, requirements manager, event server and subscriber manager. The requirements manager manages the requirements and publishes the event messages to the event server, whenever a change request occurs. The event server is responsible for three main activities: (i) It handles subscriptions from dependent objects. (ii) It is responsible for listening to event messages received from the requirements manager. (iii) It publishes or forwards event messages to the relevant subscribers. The subscriber manager is responsible for listening to the event server and for receiving and managing the event notifications.

Event-based traceability handles functional requirements and non-functional requirements, and it provides a solution to the traceability update problem. Additionally, it can be used in combination with requirements management tools like DOORS and Rational RequisitePro according to [16].

Information retrieval. The information retrieval [20, 44, 21] approach (IR) is used to automate the generation of traceability links. Commonly used IR methods include: Vector space model (VSM), different probabilistic models and latent semantic indexing (LSI) [58]. Information retrieval methods are based on similarity comparison and probabilistic values of two artifacts.

According to [68] IR methods include three general steps: (i) Pre-processing. (ii) Analyzing, indexing, creating its representation and archiving. (iii) Analyzing incoming artifacts using some ranking algorithms.

Whenever a pair of artifacts reach a specific rank, it is considered as a candidate link that must be reported to the analyst for a final decision, i.e. the analyst differentiates between true and false links. IR methods significantly reduce the effort required for creating traceability links between artifacts but, on the other hand, it still requires significant efforts by the analyst.

IR methods like VSM and LSI are used in the RT tool RETRO [57, 45]. Research on IR methods has focused on tracing functional requirements, however there is one exception where the authors in [14] used IR methods in goal-centric traceability for tracing non-functional requirements (this is covered later in this section).

Rule-based approach. The basic purpose of the rule-based (RB) approach is to automatically generate traceability links using rules [76]. There are two traceability rules, i.e. requirement-to-object-model traceability (RTOM) rule and inter-requirements traceability (IREQ) rule.

These rules are used for three specific documents: Requirements statement document (RSD), use case documents (UCD), and the analysis object model (AOM). RSD and UCD are traced to an AOM by using RTOM rules. IREQ rules are used for tracing between RSD and UCD. The RB approach presents all of the documents and both of the rules in an XML-based format. The RB approach consists of four steps: Grammatical tagging of the artifacts, converting the tagged artifacts into XML representations, generating traceability relations between artifacts, and generating traceability relations between different parts of the artifacts.

Feature-model based approach. The feature-model based (FB) approach is described in [67]. In feature modeling, requirements are described as overviews and models as a variability of the product line. A feature model consists of graphs along with nodes and edges, while nodes are the features and edges are the features relations. Each feature represents a property of the product from a customer's point of view. There are three types of features: Functional features, interface features, and parameter features. Feature relations can be classified into three categories: Hierarchical relations; in this case most important feature is placed at a high position in the hierarchy. Refinement relations, which are used to define the relations of generalization, specialization and aggregation. Exclude relations, which define the constraints between variable features which can influence the sequence of the decision of the product.

Scenario-based approach. According to the proposal by [14, 68] scenarios are used to model system functionality and to generate functional test cases. Scenario-based test cases create a mapping between requirements and other artifacts like design and code. The traceability is established by mapping scenarios with the design elements. Scenarios are created to trace only the interesting cases therefore they might not provide complete coverage. However, scenarios are frequently used by

several architectural assessment methods like the architectural trade-off assessment method and the software architecture assessment method.

Process centered engineering environments. In [62], the authors describe a traceability technique, i.e. process centered engineering environment (PCEE). The technique is used for tracing non-functional requirements. A PCEE is composed of three domains: Modeling, enactment and performance. Processes and traceability tasks are defined in modeling domain. The software engineering process and related traceability tasks are controlled by the enactment domain. These software engineering and traceability tasks are implemented in the performance domain.

The process centered engineering environments can be used to trace both functional and non-functional requirements [14, 62]. Non-functional requirements can be traced by connecting them with architectural assessment methods in the enactment domain.

Design patterns. In [40], the authors propose a technique using design patterns for tracing non-functional requirements. The technique was utilized by [14] in a model to depict traceability links between a soft-goal interdependency graph and underlying object-oriented design. This model is based on the application of pattern detection algorithms within a subset of high-level explicitly traced classes. The technique supports traceability of any non-functional requirement that can be implemented as a design pattern.

Traceability matrices. Traceability matrices are often used in industry to define relationships between requirements and other artifacts [14], e.g. design modules, code modules and test cases. By using traceability matrices the links, between requirements and other artifacts, are often manually created. Traceability matrices suffer from scalability and maintenance problems according to [14].

Keywords and ontology. In [14], a technique is described named "keywords and ontology", which is based on language extended lexicon [24].

Keywords and ontology provides traceability support between UML diagrams and non-functional requirements modeled in the form of goal tree. Keyword representing both domain and non-functional requirements, are embedded in a goal tree and UML diagrams. This approach requires maintenance and systematic use of keywords throughout the evolution of system; therefore there is no need to maintain a central traceability matrix.

Aspect weaving. Aspect oriented programming (AOP) establishes traceability between aspects (lower level NFR) and code. According to [14] in AOP suitable concerns are modeled as aspects. In these aspects dispersed functionality is encapsulated into a single entity. This single entity is woven into the code with the help of a special complier based on the set of aspect weaving rules.

Concerns can be categorized as functional and non-functional [14]. The non-functional concerns include high level NFR such as maintainability, performance and

security. Non-functional concerns are less concrete to be implemented as aspects. Functional concerns are more concrete concerns because their behavior is easily definable and they can be expressed in terms of aspect weaving rules. The functional concerns include requirements like e.g. logging and authentication.

Goal-centric traceability. The traceability of NFRs (non-functional requirements) is difficult. This is due to the fact that extensive interdependencies and trade-offs exist between them. In [20], the authors propose a technique called goal-centric traceability (GCT) to trace non-functional requirements. This technique has been successfully evaluated in a case study on a real world project.

In GCT a soft-goal interdependency graph (SIG) is used to model NFRs as goals and operationalizations [20]. The SIG is a framework which helps developers to model NFR during software development. In a SIG goals are the NFRs to be satisfied, whereas operationalizations are development, design or implementation techniques that help in satisfying NFRs [13].

GCT has four distinct phases [20]: Goal modeling, impact detection, goal analysis and decision making. These phases are briefly explained below.

Goal modeling occurs during elicitation, specification and architectural design of the system. Goal modeling is sub-divided into two phases, developing a SIG and maintaining SIGs. When developing a SIG, the non-functional goals are modeled as soft-goals, which are decomposed into operationalizations and then negotiated and agreed with the stakeholders. During the maintenance phase a SIG is maintained throughout the life of software system to accommodate the impact of change. Traceability links are established as functional model of the system and set of potentially impacted SIG elements. One representation of the functional model can be UML diagrams. The functional change is usually implemented at code level or design level.

The ability to understand the impact of change in UML models help developers to evaluate the impact of change before implementing it in the code.

The third phase, the goal analysis, is sub-divided into two sub-phases. During contribution re-analysis the impact of change is propagated throughout the SIG to evaluate its effects on system wide goals. While, according to [20], during goal re-evaluation each operationalization is examined to determine how the proposed change influences the satisfaction of its parent goal. Any parent goal that no longer satisfied is also re-evaluated to determine how it impacts its own parent. This process is continued until all potential goals have been re-evaluated. The output of this phase is an impact analysis report that identifies all goals that are either positively or negatively affected by the proposed change.

Finally, the decision making phase is sub-divided into two sub-phases: Decision and impact evaluation. During the decision sub-phase stakeholders examine the impact report to decide whether to proceed with the proposed change or not. In the impact evaluation sub-phase stakeholders evaluate the impact of the proposed change upon NFR goals and identify risk mitigation strategies.

Concerning empirical evidence, the authors in [20] report on an experimental evaluation of GCT in the Ice Breaker System. The system manages de-icing services to prevent ice formation on roads. This system receives information from weather stations and road servers. This system consists of 180 functional requirements and nine NFR related to accuracy, availability, safety, usability, security, extensibility, completeness and cost.

The results of the case study reveal that GCT provides support to developers to manage the impact of functional changes on non-functional requirements. The use of a probabilistic retrieval algorithm reduces tracing efforts in link retrieval. The probabilistic retrieval algorithm dynamically retrieves traceability links for NFRs.

3.4.1. *Analysis of requirements traceability techniques*

On the basis of extracted information we classify RT techniques on the rationale of requirements type (functional, non-functional or both) and traceability aspects.

Value-based and feature-oriented requirements traceability techniques are used to trace functional requirements. Whereas keywords and ontology, design pattern, and goal-centric traceability techniques provides traceability for non-functional requirements.

The techniques which provide traceability for both functional and non-functional requirement are pre-RS requirements traceability, event-based traceability, information retrieval, traceability matrices, process centered engineering environment, aspect weaving, and scenario-, hypertext- and rule-based approaches.

Among all these traceability techniques only pre-RS requirements traceability technique emphasis on pre-RS traceability aspects, the rest of the techniques focus on post-RS traceability aspects. The results of this discussion are summarized in Table 13.

It is obvious from Table 13 that 15 techniques were identified in the systematic review, however it is interesting to note that almost three out of four lack empirical evidence.

Similarly it can be observed from Table 13 that some techniques are discussed in more than one paper. In Fig. 3 one can see how often a technique has been 'published', which might indicate how mature a techniques is.

Among the techniques shown in Fig. 3 only information retrieval and goal-centric traceability have empirical evidence; whereas techniques like event-based, rule-based and scenario-based traceability have no empirical evidence reported in literature. To be blunt, event-based traceability is discussed in nine articles and but has no reported empirical evidence, i.e. experiment, survey or case study, that is reported in the articles.

In [16], M-Net, a web based conferencing system with 300 requirements and an initial set of 250 links, is analyzed. The example was used to illustrate the concept of event-based traceability. Unfortunately, this example only demonstrated the feasibility of event-based traceability to solve synchronization problems related to the

Table 13. Summary of comparison between different traceability techniques. (F and NF, in the fourth column from left, stands for functional and non-functional, respectively.)

Ref.	Technique	Aspect	Type of req.	Empirical evidence
[3]	Feature-oriented requirement tracing	Post-RS	F	Case study
[22, 14, 68] [19, 17, 69] [44, 16, 18]	Event-based traceability	Post-RS	F & NF	None
[14]	Matrices	Post-RS	F & NF	None
[14]	Keywords & ontology	Post-RS	NF	None
[14]	Aspect weaving	Post-RS	F & NF	None
[14, 20]	Goal-centric	Post-RS	NF	Experiment
[14, 68]	Scenario-based	Post-RS	F & NF	None
[20, 14, 68] [57, 25, 44] [21, 4]	Information retrieval	Post-RS	F & NF	Case study
[40]	Design patterns	Post-RS	NF	None
[47]	Value-based requirements traceability	Post-RS	F	Case study
[66]	Pre-RS requirement tracing	Pre-RS	F & NF	None
[68]	Hypertext-based	Post-RS	F & NF	None
[68]	Feature-model based	Post-RS	F & NF	None
[68, 76]	Rule-based	Post-RS	F & NF	None
[62]	Process centered engineering environment	Post-RS	F & NF	None

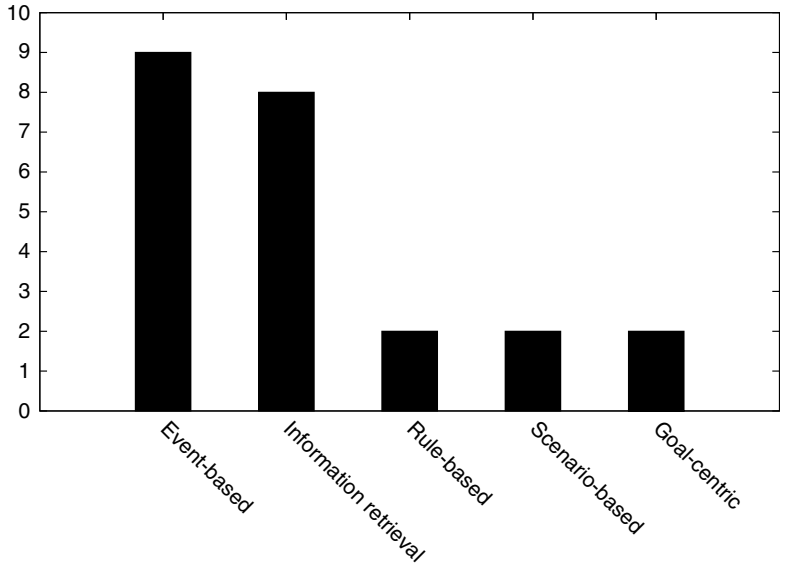

Fig. 3. Descriptive statistics showing the number of times a technique has been 'published' according to the primary studies.

Table 14. Conclusions drawn from the empirical evidence on traceability techniques.

Refs.	Technique	Conclusions
[3]	Feature-oriented requirements tracing (FORT)	(1) FORT provides variability information based of feature modeling which is useful to estimate requirements change. (2) FORT reduces efforts by being based on feature prioritization.
[14, 20]	Goal-centric traceability (GCT)	(1) GCT facilitates to manage impact of functional change upon the non-functional requirements. (2) GCT helps to manage critical system qualities such as safety, security, reliability, usability and performance.
[20, 14, 68] [57, 25, 44] [21, 4]	Information retrieval (IR)	(1) It is very hard to maintain links in constantly evolving systems. (IR methods facilitate dynamic link generation.) (2) The results of a case study indicate that "IR provides a practicable solution to the problem of semi-automatically recovering traceability links between code and documentation".
[47]	Value-based requirements tracing (VBRT)	(1) The traceability efforts are reduced by focusing on most important requirements as compared to full tracing. (2) It is easier to capture traceability related information in earlier phases of software development lifecycle. (3) The prioritization step of VBRT identifies important requirements to be traced in more detail than others. (4) In VBRT important requirements are identified based on parameters stakeholder value, requirements risk/volatility and tracing costs. (5) Tracing requirements into code at method level provides more useful and detailed information than tracing into class level.

updating artifacts and resolving traceability links. (The authors claim that a long-term empirical study of event-based traceability is currently under way.)

In [22] the authors describe a framework known as TraCS. It uses event-based traceability to trace performance related requirements, however there is no empirical evidence connected to TraCS.

The conclusions drawn from examining the empirical evidence of RT are reported in Table 14. These conclusions could be of interest while selecting a particular technique for providing solutions in academia or industry or when developing a framework for requirements traceability.

Based on the systematic review results, the traceability techniques can be divided into two types:

(1) Techniques facilitating pre-RS traceability. This type includes those traceability techniques which help to describe the life of requirements when they are not included in the requirements specification. There is only one technique in this category, i.e. pre-RS requirements traceability (see Sec. 3.4).
(2) Techniques facilitating post-RS traceability. This type includes those techniques which help to trace the life of requirements when they are included in the

requirements specification and forward. These techniques can be further divided into three types based on the systematic review's results (see Table 8).

(a) Techniques focusing on tracing functional requirements, i.e. VBRT and FORT.
(b) Techniques focusing on tracing non-functional requirements, i.e. design patterns, keywords and ontology, and goal-centric traceability.
(c) Techniques favoring traceability of both functional and non-functional requirements. The techniques included in this category are EBT, IR, hypertext-, featuremodel- and scenario-based approach, process centric environment, matrices and aspect weaving.

An interesting future possibility would be to investigate if traceability links can be mined and found by using different types of similarity metrics on the text level as done for tests in [33].

4. Answers to Questions Connected to the Systematic Review

In this section the questions presented in the review protocol (see Sec. 2.1) are addressed. By answering these questions the two research questions will also be answered.

Question 1. What is requirements traceability based on state of the art research and standards?

The definition of RT based on state of the art research and standards is discussed in Sec. 3.1. It is obvious from the systematic review results that Gotel and Finkelstein, Hamilton and Beeby, and IEEE Std. 830−1998 provide comprehensive and state-of-the art definitions of RT, and that the definition of Gotel and Finkelstein is most widely used in literature. Hence according to Gotel and Finkelstein's view, RT is the ability to follow and describe the life of requirements both in forward and backward direction throughout the development, deployment and refinement cycle.

Question 2. Which are the challenges when implementing requirements traceability and how does research address these challenges?

The challenges when implementing RT (and possible solutions) are discussed in Sec. 3.2.

We have classified these challenges into three categories: (i) Challenges addressed by academia (Table 5). (ii) Challenges addressed by industry (Table 6). (iii) Challenges addressed by neither academia nor industry (as described in 3.2.1). The general challenges of cost and value of RT, i.e. the projects have the costs, but both pre- and post-project phases benefit from well realized and maintained RT, may be central.

Question 3. Which are the various requirements traceability tools according to research literature?

The RT tools identified by the systematic review are discussed in Sec. 3.3. The authors have categorized RT tools into three categories as shown in Fig. 4. These

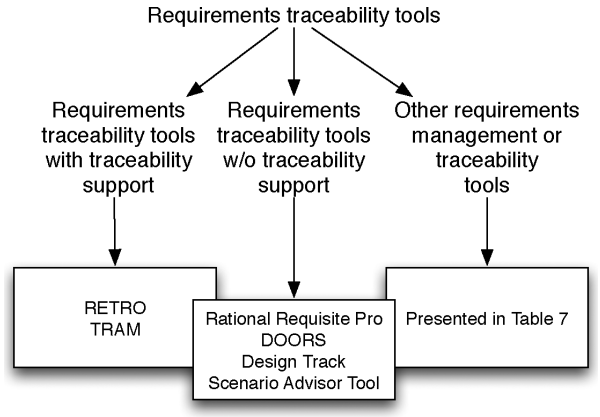

Fig. 4. Requirements traceability tools as found by this systematic review.

categories are requirements management tools providing traceability support, RT tools and other requirements management/traceability tools. The lack of empirical evaluation (validation) of the tools might be a serious indicator.

Question 4. What requirements traceability techniques are reported in research literature?

The RT techniques reported in the primary studies are discussed in Sec. 3.4. We have grouped traceability according to two aspects of RT as shown in Fig. 5. One interesting finding, which is noticeable through this categorization, is the fact that there is very little research done in the area of pre-RS traceability. In addition it should be noticed that 11 out of 15 RT techniques lack empirical validation either through lab-validation in academia (e.g. experiments), or through industry case studies or pilots.

5. Interviews Conducted in Industry

In order to determine RT challenges in industry, interviews were conducted with two software development companies. These interviews are summarized below. The questionnaire used and the response to the questions can be found at [78].

5.1. *Company A*

5.1.1. *Introduction*

Company *A* is based in Sweden and a world-leading telecommunication company (the interviewee were selected from one of the company's divisions). Company *A* provides telecommunications equipment and other telecom related services in 140 countries. More than 1,000 networks use Company *A*'s products and 40% of all mobile calls are made through systems provided by Company *A*. Company *A* is one of few telecom companies that offer end-to-end solutions for all major mobile communication standards. The company is ISO 9000 certified.

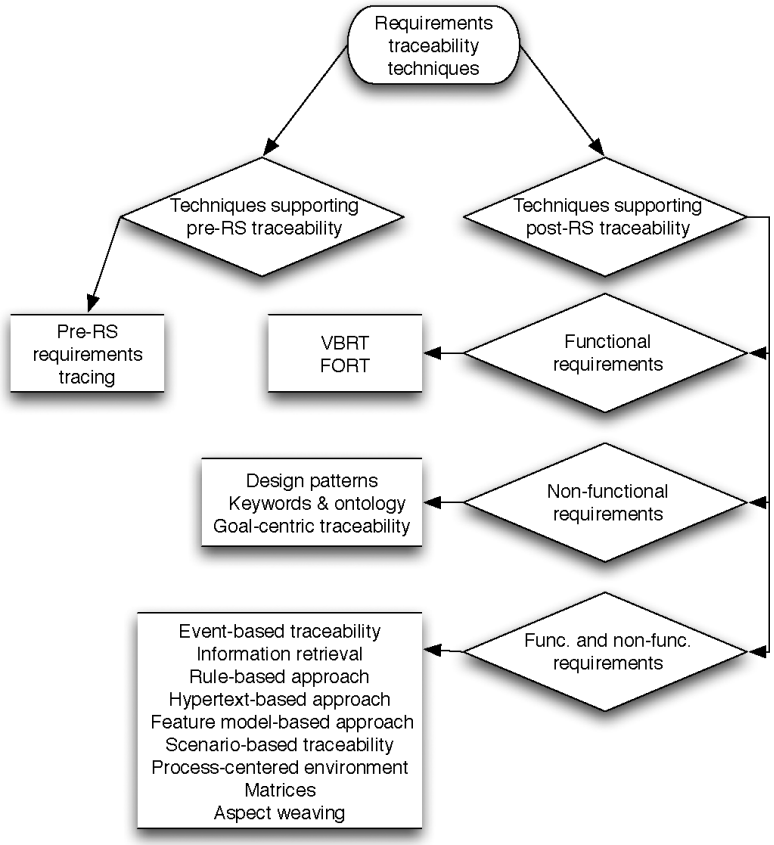

Fig. 5. Requirements traceability techniques found by the systematic review.

5.1.2. *Interviewee*

The authors' first interviewee was person x. She is working as a discipline driver for a whole product unit. She is also responsible for the implementation of RT practices. Person x has been working at Company A for the last ten years and she has specifically worked with RT for the last five years. The authors' second interviewee was person y. Person y is working as a requirements manager and responsible for overall requirement engineering activities. Person y has been working at Company A for the last nine years and focused on RT for the last 1.5 years.

5.1.3. *Traceability practices*

Persons x any y believe that RT is an important and useful activity within requirements management. There are thirty people working in requirements engineering and 3−5 people are dedicated to RT practices. According to the interviewees traceability is beneficial for the company due to several reasons.

One of the major benefits is customer satisfaction. Customer can check and follow the product development progress according to their requirement. Requirements traceability provides a guarantee for requirements engineers and product managers that every requirement is implemented. Requirements traceability helps to ensure impact analysis and derivation analysis. Requirements traceability facilitates impact analysis. Whenever a change request is initiated at any stage we can trace the requirements or design artifacts or test cases that can be affected. Similarly in coverage analysis RT ensures implementation of all requirements. Interviewees also mentioned that normally there are 200 change requests per project.

5.1.4. *Tool support for traceability*

According to x and y, Company A use an automated requirements management tool named MARS which is developed by IBM specifically for Company A. There is a specific module in MARS which is responsible for traceability. Company A also uses another tool named 'Focal Point' for storing the elicited requirements. After entering the requirements into Focal Point the Main Requirement Specification (MRS) is generated from it. After generating the MRS, it is entered into MARS.

By using MARS, requirements are stored in a central repository. Each requirement has a number of attributes and a unique identifier, i.e. ID, slogan and description. Traceability of requirements is obtained by using MARS and Focal Point as both tools are synchronized. Requirements traceability matrices are generated to cater for RT. These matrices are used to constantly be up-to-date regarding the status of a requirement and whether the requirement is implemented or not. MARS also facilitates change management. Change requests are initiated based on the requirements stored in MARS. A change control board analyzes the change request and decides to accept or reject it. In case of acceptance, changes are made permanent in MARS (versioning is also handled).

5.1.5. *Factors influencing requirements traceability*

According to the interviewees traceability is very well implemented in MARS. Despite this, there are a few issues e.g. the manual decomposition of master requirements specification into detailed requirements for RT. Furthermore the maintenance cost of MARS is also very high, therefore Company A is thinking about using another tool in the future.

5.2. *Company B*

5.2.1. *Introduction*

Company B develops both bespoke and market-driven products for its customers. This company is involved in developing application suites and frameworks for mobile

phones based on the Symbian operating system. Company *B* is neither ISO9000 nor CMMI certified.

In Company *B* the requirements engineering activities are monitored by a department named Software Development Organization (SDO). The responsibility of SDO is to capture, clarify, process and trace requirements to testing and implementation. There are thirty people working in the SDO; out of which 13 people work with capturing requirements.

5.2.2. *Interviewee*

Interviewee *z* has been working as a software quality engineer in Company *B* for the last two years. The current job responsibilities of person *z* include process development, requirements matrix measurement and controlling the development projects.

Although person *z* is not directly working with requirements engineering but as a software quality engineer, he has knowledge of requirements engineering activities in Company *B*.

5.2.3. *Traceability practices*

In Company *B* the high level requirements are known as Market Requirement Specification (MRS). They are decomposed into Product REQuirements (PREQs). The PREQs are further decomposed into a Feature Specification (FS), which is specified at a low level. Traceability is maintained by establishing connection between every MRS, PREQ and FS; and *vice versa*.

There is no traceability in design and source code. In testing, each FS is connected to at least one test case which must have passed. The traceability is only maintained within the requirements and testing phases.

5.2.4. *Tool support for traceability*

Company *B* is using two different tools for managing requirements and test cases. These tools are Requirements Management System (RMS) and Test Management System (TMS). The RMS is used to manage requirements like MRS, PREQ and FS. It also provides traceability support but only against the requirements part. TMS is used to manage and store test cases.

There is no link between the RMS and the TMS. The traceability against FS, to test cases, is maintained manually with the help of a Requirements Traceability Matrix (RTM). The RTM is maintained in MS Excel.

5.2.5. *Benefits of implementing traceability*

In Company *B*, requirements are decomposed into Market Requirements Specification (MRS) which is further decomposed into PREQ and then to FS. Traceability provides a higher level of details for the software engineers and helps in identifying

the source of the FS. Similarly traceability from MRS to test cases ensures that the feature is 'complete'.

Traceability helps in impact analysis. In Company *B* impact analysis is done in several ways. If it is done early in the development life-cycle during PREQ phase; usually by the product manager. The product manager should have sufficient competency to declare that this change will affect the product in one way or another. But during development, impact analysis is done by the team responsible for managing the component affected by the change.

5.2.6. *Factors influencing requirements traceability*

Company *B* has good traceability support between feature specifications, product requirement specifications and functional specification according to person *z*. But at the moment it is very difficult to determine requirements completion in a project. The requirement completion means that the FS has been developed and verified for specific project. There is no direct link between RMS and TMS. The FS that is verified for a particular project is then maintained manually using MS Excel. Therefore, tool support is one factor affecting RT.

Company *B* is not maintaining traceability between requirements to design and code phase. The traceability is maintained only between requirements and test cases. Therefore it is very difficult to identify the components that are affected by change requests in design and development phase. There are normally 5−8 change requests for every twenty features.

5.3. *Analysis of interviews*

The results of the case studies are interesting in the sense that Company *A* and Company *B* are fairly different with regards to traceability policies. The analysis of the interview reveals that they can be placed into two different categories [63], i.e. high-end and low-end traceability users.

For requirements management, Company *A* is using a formal tool, MARS, which also provides traceability support. MARS was tailored by IBM for Company *A*. On the other hand Company *B* lacks formal methods for traceability. Company *B* uses automated tools for requirements management (RMS) and test case management (TMS). Although the automated traceability is only maintained in RMS. There is no traceability in design and code. Furthermore, to establish traceability links between RMS and TMS, traceability matrices are manually maintained using MS excel.

Company *B* uses static methods like traceability matrices that are not updated automatically as the system evolves. Therefore, manual traceability matrices lose most of their usefulness after creation (low-end traceability users often exhibit this aspect). Whereas, high-end traceability users recognize the need for maintaining traceability, which helps in reflecting the current status of the system and generate traceability documentation at any point in the development life-cycle. This practice is exercised by Company *A* and not by Company *B*.

As stated by [63], low-end traceability users view traceability as a mandate from the customer, while high-end traceability users consider traceability as an important component of the overall quality engineering process. In Company *A* traceability is an integral part of the quality assurance process. Similarly there are approximately 200 change request per project in Company *A* and, hence, they rely on traceability to identify the affected software artifacts due to change request. On the other hand, their automated requirements management tool (MARS) also provides richer traceability support.

In short, Company *B* implements traceability to ensure that two conditions are fulfilled. First, that high level requirements known as MRS are decomposed into low level requirements called Feature Specifications. Second, to guarantee that a feature is complete. In the practices followed by Company *B*, a feature is said to be complete when there is a traceable link between MRS and test cases. Albeit these conditions are not enough by themselves, we can conclude that Company *A* has a fairly well defined traceability processes. In the future Company *A* is planning to shift to another requirements management tool (Rational RequisitePro), a tool which is briefly discussed in Sec. 3.3. One question is if a new tool will solve the problems with high maintenance identified as a major challenge for Company *A*?

In Company *B* traceability practices are not fully implemented. Although they are using tools for managing requirements (RMS) and tools for managing test cases (TMS), these tools provide traceability only for requirements and test cases, i.e. there is no traceability support between RMS and TMS. Similarly no traceability information is maintained for traceability between the design and development phase.

High-end traceability users like Company *A* view traceability as one of the core requirements engineering practice. They believe that traceability delivers more than it costs. They can develop traceability tools in-house or purchase them. On the other hand, low-end traceability users like Company *B*, while acknowledging the importance of traceability, treat traceability more as an expensive overhead.

In the systematic review the authors identified the challenges related to traceability (reported in Tables 5 and 6). The views of Companies *A* and *B* are summarized in Table 15 connected to the challenges identified by the systematic review.

Table 15 contains some very interesting results. For instance both companies believe that traceability delivers more than it costs. But in case of Company *B* there is no traceability in design and code phase and, in addition, Company *B* uses manual RT for maintaining traceability links.

The challenges related to RT mapped to both the systematic review and the case studies can be seen in Table 16.

It is evident from Table 16 that some of the challenges identified by the systematic review also hold in practice.

6. Threats to Validity

An analysis of validity threats enhance the accuracy of a research design by identifying factors which can affect the results. In this section four different kinds of validity threats and their implications are discussed [82].

Table 15. Views of Company A and B concerning challenges identified in the systematic review.

Ref.	Issues identified in systematic review	View of Company A and B
[5, 9]	Requirement traceability does not offer immediate benefit to development process.	Both companies disagree with it. Requirements traceability offers benefits for both the companies.
[9, 22]	Lack of coordination between team members, developers think that traceability costs more then it delivers, excessive use of traceability generate more links that are not easy to manage.	Both companies believe that traceability delivers more than it cost. Both companies do not consider lack of coordination between team members and generation of traceability links as factors affecting traceability.
[9, 63]	Organizational, environmental and technical factors.	These factors are observed in both companies. Based on these factors Company A is classified as being a 'high-end traceability user' and Company B is classified as 'low-end traceability user'.
[16]	Informal development methods, insufficient resources, time and cost for traceability, lack of coordination between people, failure to follow standards.	Among these challenges the cost for traceability is a significant factor for both companies.
[14]	Failure to trace non-functional requirements (NFR) like security, performance and usability.	These factors are important for both companies. The NFRs like security, performance and usability are of interest to Company B as they are developing applications for mobile phones. However Company B lacks support for traceability of NFR.
[21]	Manual construction of requirements traceability matrix (RTM) is costly.	Manual construction of RTM is costly for Company A, therefore they are focusing on using automated tools. Manual construction of RTM is cheap for Company B, therefore they are using MS Excel sheets for manual construction of RTM.
[38]	Some problematic questions are identified as challenges e.g. who identify a requirement and how, who was responsible for the requirement to begin with, who is currently responsible, who will take care of change (s) in requirements, what will be the effect on project in terms of knowledge loss, if some individual or the whole group leave from company?	These challenges are not important for both companies. Company A is using the MARS tool and Company B is using RMS that takes care of these challenges.
[39]	Requirements change by user, less appropriate information is available for making decision with requirements.	Requirements change by user is always there but both companies have maintained enough information to make decision about requirements.
[47]	Cost and effort related to RT.	Cost related to RT is important issue for both companies.
[66]	Problems associated with tracing back to their sources.	This issue is resolved in Company A by using the MARS tool. This problem is important but unsolved in the case of Company B.

Table 15. (*Continued*)

Ref.	Issues identified in systematic review	View of Company *A* and *B*
[80]	Requirements management challenges in industry projects like e.g. inadequate impact analysis, lack of information transfer.	Both companies have resolved these challenges by requirements management tools like MARS and RMS.

Table 16. Most important challenges/issues deduced from the systematic review and from interviews in industry. In the third column SR is short for systematic review and I is short for interview conducted in industry.

Problem/issue	Motivation	Source
Cost and efforts related to RT.	Cost and efforts are the major factors due to which companies do not use traceability.	SR & I
How much traceability is enough?	This question is still unanswered and research community is working to solve this issue.	SR
Change requests or requirements change.	If a company is not following traceability practices then it is very difficult to manage change requests.	SR
Failure to trace non-functional requirements.	There are number of inter-dependencies and trade-offs between non-functional requirements. Therefore NFR are important with respect to traceability.	SR
Tracing requirements back to their sources.	Tracing requirements back to their sources helps to identify origin of requirements. This issue is one of the key factors responsible for defective RT.	SR
Traceability support between all phases of a software development life-cycle.	Traceability should be maintained between all the artifacts produced during the software development and maintenance. In e.g. Company *B* there is a lack of traceability support design and code phase.	I

6.1. *Conclusion validity*

One of the main purposes of a review protocol in a systematic review is to eliminate researcher bias [55]. The review protocol (see Sec. 2.1) was reviewed by two independent researchers having experience in conducting systematic reviews. In addition, the relevance of search terms and search resources defined in the review protocol was ensured by conducting pilot searches.

The questionnaires used for the interviews in industry were validated to rectify poor questions and flow in the layout of questionnaire. The heterogeneity of subjects is another threat to conclusion validity [82]. Heterogeneity of subjects means that the subjects belong to varied group with respect to background, education and experience. While homogeneity of subjects means that the subjects belong to the same group based on education, background and experience. In our case, person x worked as a discipline driver in the process area of RT for five years whereas person y worked as a requirements manager and had experience 1.5 years of experience in RT. Person z had been working as a software quality engineer for two years. In short, the subjects of our interviews would be hard to categorize as heterogeneous or homogeneous.

6.2. *Construct validity*

Evaluation apprehension is the main threat to construct validity. According to [82], evaluation apprehension means that humans have the tendency to look better when they are evaluated. On the other hand some people are afraid of being evaluated. In order to eliminate this threat the interviewees in both companies were ensured that they and their companies would be anonymous.

Mono-operation bias is another threat to construct validity. Mono-operation bias means that there is a single independent variable, case, subject or treatment in a research study [82]. We have decreased this threat by conducting interviews at two software companies. In Company A we interviewed two requirements engineers and in Company B we interviewed one quality engineer.

6.3. *Internal validity*

In our case, the interviews were recorded and it is a common observation that people may not feel comfortable if the interview is audio taped. In order to eliminate this potential threat to internal validity the interviewees were assured that the recordings would only be used by us.

The internal validity threat related to the systematic review is mainly the publication bias. Publication bias refers to the fact that positive results are more likely to be published than the negative facts [55]. In order to eliminate this threat the selected conferences and proceedings mentioned in Table 1 were manually searched. Similarly the 'study quality assessment and procedures', as mentioned in the review protocol, decreased the factor of publication bias according to our opinion.

Selection of subjects from a population also affects internal validity [82]. In our case we interviewed people working in the process area of requirements engineering. However, one of our subjects (person z) did not work directly with RT. Nevertheless, person z had adequate educational background, experience and knowledge about the practices related to RT and requirements engineering carried out in Company B.

6.4. *External validity*

One of the threats to external validity is the interaction of selection and treatment [82]. This occurs when wrong subjects are taken from the population and, hence, the results cannot be generalized to the whole population. In our case the subjects of our interviews are people working in the process area of RT. On the other hand, we acknowledge, that there still is a threat that validation results may not be generalized to companies related to other software engineering domains.

An additional threat to validity in this case is our selection of two Swedish companies for the validation. We acknowledge that threat but would also like to stress that the purpose of the validation was to do a limited validation of the findings (or dynamic evaluation as described in [35]).

7. Conclusions

This paper presented a systematic literature review on requirements traceability (RT). The systematic review augmented and was to parts validated in industry through two case studies.

The first thing evaluated by the systematic review was to ascertain the most commonly used definition of RT. Gotel and Finkelstein's, "the ability to describe and follow the life of a requirement in both forwards and backwards direction (i.e. from its origins, through its development and specification to its, subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases)", was the most commonly used definition (about 80%). The main motivation for this investigation was to ascertain what common understanding researchers working on RT had.

The systematic review maps to both the challenges identified, and the techniques and methods developed to address them. An interesting observation is that cost is often used as a main motivation for not implementing adequate traceability policies, and maintaining traceability. Large parts of the costs of maintaining traceability rests on the project, and while the benefits are felt by projects, they are also spread to both pre- and post-project activities. Given this, project motivation may be less than optimal, possibly influencing or enhancing other issues such as coordination, roles and responsibilities, and the use of techniques.

Looking at both the techniques/methods and tools developed to address the challenges, additional observations can be made. First, most techniques and tools were not validated empirically. Even the ones covered in several publications lacked validation, the papers focused on extensions and new application, not evaluation. This might be a result of not catching the validation publications in the review, although we consider the sheer amount to be an indicator of overall validation being absent in many cases. The implications from a academic perspective is that we build on non-validated techniques as we refine and extend, and from a industrial perspective, with no empirical evidence it is hard to gauge the usability and usefulness beyond the illustrations offered in the publications.

Among all traceability techniques only the pre-RS technique emphasizes pre-RS traceability aspects, the rest of the techniques focus on post-RS traceability aspects. This might be an indication of that development projects are the focus, and not what happens before or after a project. The possible implications of this is evident as the benefits of RT is only partly associated with the project. This is especially true looking at product development going beyond the 'one shot' bespoke development projects.

The realization that RT reaches over the project perspective and into the product life-cycle perspective might be an important realization for both industry and researchers. It will enable techniques that offer a complete RT solution, which might include, and take inspiration from, the good-examples presented in this systematic review, and extending them to cover a life-cycle perspective. These new techniques would also give industry the possibility to see RT as something more than a project problem as the cost and effort can additionally be spread over the pre-RT (e.g. product management) and post-project (next release or maintenance) perspectives.

References

1. A. Abran, J. W. Moore, P. Bourque and R. Dupuis (editors), *Guide to the Software Engineering Body of Knowledge* (*SWEBOK*) (IEEE Computer Society, Los Alamitos, 2004).
2. W. Afzal, R. Torkar and R. Feldt, A systematic review of search-based testing for non-functional system properties, *Information and Software Technology* **51**(6) (2009) 957−976.
3. S. Ahn and K. Chong, A feature-oriented requirements tracing method: A study of cost-benefit analysis, in *Proc. of 2006 International Conference on Hybrid Information Technology*, Washington, DC, USA, 2006, IEEE Computer Society, pp. 611−616.
4. G. Antoniol, G. Canfora, G. Casazza, A. De Lucia and E. Merlo, Recovering traceability links between code and documentation, *IEEE Transactions on Software Engineering* **28**(10) (2002) 970−983.
5. P. Arkley and S. Riddle, Overcoming the traceability benefit problem, in *Proc. of the 13th IEEE International Conference on Requirements Engineering*, Washington, DC, 2005, pp. 385−389.
6. A. Aurum and C. Wohlin, *Engineering and Managing Software Requirements* (Springer-Verlag, New York, 2005).
7. S. C. Bailin, A. M. Davis, M. Dorfman, R. E. Fairley, S. R. Faulk, K. Forsberg, L. M. Ippolito, H. Mooz, J. D. Palmer, J. Reilly, H. Saiedian, R. H. Thayer and D. R. Wallace, *Software Requirements Engineering*, 2nd edition (IEEE Computer Society Press, Los Alamitos, 1997).
8. M. F. Bashir and M. A. Qadir, Traceability techniques: A critical study, in *IEEE Multitopic Conference*, Washington, DC, December 2006, pp. 265−268.
9. F. Blaauboer, K. Sikkel and M. N. Aydin, Deciding to adopt requirements traceability in practice, in J. Krogstie, A. L. Opdahl and G. Sindre (eds.), *CAiSE*, LNCS (Springer, 2007), pp. 294−308.
10. F. Bouquet, E. Jaffuel, B. Legeard, F. Peureux and M. Utting, Requirements traceability in automated test generation: Application to smart card software validation, *SIGSOFT Software Engineering Notes* **30**(4) (2005) 1−7.

11. F. Bubl and M. Balser, Tracing cross-cutting requirements via context-based constraints, in *Proceedings of the 9th European Conference on Software Maintenance and Reengineering*, Washington, DC, USA, 2005, pp. 80−90.
12. P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell and J. Natt och Dag, An industrial survey of requirements interdependencies in software product release planning, in *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, Washington, DC, 2001, p. 84.
13. L. Chung, B. A. Nixon, E. Yu and J. Mylopoulos, *Non-Functional Requirements in Software Engineering* (The Kluwer International Series in Software Engineering, Volume 5) (Kluwer Academic Publisher, 1999).
14. J. Cleland-Huang, Toward improved traceability of non-functional requirements, in *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering* (ACM, New York, NY, USA, 2005), pp. 14−19.
15. J. Cleland-Huang, B. Berenbach, S. Clark, R. Settimi and E. Romanova, Best practices for automated traceability, *IEEE Computer* **40**(6) (2007) 27−35.
16. J. Cleland-Huang, C. K. Chang and M. Christensen, Event-based traceability for managing evolutionary change, *IEEE Transactions on Software Engineering* **29**(9) (2003) 796−810.
17. J. Cleland-Huang, C. K. Chang and Y. Ge, Supporting event based traceability through high-level recognition of change events, in *Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment*, Washington, DC, 2002, IEEE Computer Society, pp. 595−602.
18. J. Cleland-Huang, C. K. Chang, G. Sethi, K. Javvaji, H. Hu and J. Xia. Automating speculative queries through event-based requirements traceability, in *Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*, Washington, DC, 2002, IEEE Computer Society, pp. 289−298.
19. J. Cleland-Huang, C. K. Chang and J. C. Wise, Automating performance-related impact analysis through event based traceability, *Requirements Engineering* **8**(3) (2003) 171−182.
20. J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya and S. Christina, Goal-centric traceability for managing non-functional requirements, in *Proceedings of the 27th International Conference on Software Engineering*, ACM, New York, NY, USA, 2005, pp. 362−371.
21. J. Cleland-Huang, R. Settimi, C. Duan and X. Zou, Utilizing supporting evidence to improve dynamic requirements traceability, in *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, Washington, DC, USA, 2005, IEEE Computer Society, pp. 135−144.
22. J. Cleland-Huang, G. Zemont and W. Lukasik, A heterogeneous solution for improving the return on investment of requirements traceability, in *Proceedings of the 12th IEEE International Requirements Engineering Conference*, Washington, DC, 2004, IEEE Computer Society, pp. 230−239.
23. Software Engineering Institute | Carnegie Mellon, http://www.sei.cmu.edu/cmmi/ October 2008.
24. L. M. Cysneiros and J. C. S. do Prado Leite, Nonfunctional requirements: From elicitation to conceptual models, *IEEE Transactions on Software Engineering* **30**(5) (2004) 328−350.
25. A. Dekhtyar, J. H. Hayes and S. K. Sundaram, Advancing candidate link generation for requirements tracing: The study of methods, *IEEE Transactions on Software Engineering* **32**(1) (2006) 4−19.

26. Department of Defence, US. Military standard: Defense system software development (DOD-STD-2167A), Technical report, Space and Naval Warfare Systems Command, Washington, DC, February 1988.
27. J. Dick, Design traceability, *IEEE Software* **22**(6) (2005) 14−16.
28. J. C. S. do Prado Leite and K. K. Breitman, Experiences using scenarios to enhance traceability, in *2nd International Workshop on Traceability in Emerging Forms of Software Engineering in Conjunction with the 18th IEEE International Conference on Automated Software Engineering*, Montreal, Canada, October 2003, pp. 63−70.
29. T. Dybå, V. B. Kampenes and D. I. Sjøberg, A systematic review of statistical power in software engineering experiments, *Information and Software Technology* **48**(8) (2006) 745−755.
30. M. Edwards and S. L. Howell, A Methodology for system requirements specification and traceability for large real-time complex systems, Technical report, Naval Surface Warfare Center, 1991.
31. A. Egyed and P. Grünbacher, Automating requirements traceability: Beyond the record and replay paradigm, in *Proceedings of the 17th IEEE International Conference on Automated Software Engineering*, Washington, DC, USA, 2002, p. 163.
32. E. Engström, P. Runeson and M. Skoglund, A systematic review on regression test selection techniques, *Information and Software Technology* **52**(1) (2010) 14−30.
33. R. Feldt, R. Torkar, T. Gorschek and W. Afzal, Searching for cognitively diverse tests: Towards universal test diversity metrics, in *Proceedings of 1st Search-Based Software Testing Workshop* (*SBST'08*), 2008, pp. 178−186.
34. T. Gorschek, *Requirements Engineering Supporting Technical Product Management*, PhD thesis, Department of Systems and Software Engineering, Blekinge Institute of Technology, May 2006.
35. T. Gorschek, P. Garre, S. Larsson and C. Wohlin, A model for technology transfer in practice, *IEEE Software* **23**(6) (2006) 88−95.
36. T. Gorschek and C. Wohlin, Requirements abstraction model, *Requirements Engineering* **11**(1) (2005) 79−101.
37. O. Gotel and A. Finkelstein, An analysis of the requirements traceability problem, in *International Conference on Requirements Engineering*, 1994, pp. 94−101.
38. O. Gotel and A. Finkelstein, Extended requirements traceability: Results of an industrial case study, in *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, Washington, DC, 1997, p. 169.
39. O. Gotel and S. J. Morris, Crafting the requirements record with the informed use of media, in *Proceedings of the First International Workshop on Multimedia Requirements Engineering*, Washington, DC, 2006, p. 5.
40. D. Gross and E. Yu, From non-functional requirements to design through patterns, *Requirements Engineering* **6**(1) (2001) 18−36.
41. V. L. Hamilton and M. L. Beeby, Issues of traceability in integrating tools, in *Proceedings of the IEE Colloquium Tools and Techniques for Maintaining Traceability during Design*, Piscataway, NJ, USA, 1991, pp. 4/1−4/3.
42. J. Han, TRAM: A tool for requirements and architecture management, *Australian Computer Science Communincations* **23**(1) (2001) 60−68.
43. Ø. Hauge, C. Ayala and R. Conradi, Adoption of open source software in software-intensive organizations — A systematic literature review, *Information and Software Technology* **52**(11) (2010) 1133−1154.
44. J. H. Hayes, A. Dekhtyar and J. Osborne, Improving requirements tracing via information retrieval, in *Proceedings of the 11th IEEE International Conference on Requirements Engineering*, Washington, DC, 2003, p. 138.

45. J. H. Hayes, A. Dekhtyar and S. K. Sundaram, Improving after-the-fact tracing and mapping: Supporting software quality predictions, *IEEE Software* **22**(6) (2005) 30–37.
46. J. H. Hayes, A. Dekhtyar, S. K. Sundaram, E. A. Holbrook, S. Vadlamudi and A. April, Requirements tracing on target (RETRO): Improving software maintenance through traceability recovery, *Innovations in Systems and Software Engineering* **3**(3) (2007) 193–202.
47. M. Heindl and S. Biffl, A case study on value-based requirements tracing, in *Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, New York, NY, USA, 2005, pp. 60–69.
48. R. A. Howard and J. E. Matheson (eds.), *Readings on the Principles and Applications of Decision Analysis*, Strategic Decision Group, Menlo Park, California, 1984.
49. IEEE Society, IEEE recommended practice for software requirements specifications (IEEE Std. 830–1998), Technical report, IEEE Computer Society, 1998.
50. M. Ivarsson and T. Gorschek, Technology transfer decision support in requirements engineering research: A systematic review of REj, *Requirements Engineering* **14**(3) (2009) 155–175.
51. M. Jarke, Requirements tracing, *Communications of the ACM* **41**(12) (1998) 32–36.
52. W. Jirapanthong and A. Zisman, XTraQue: Traceability for product line systems, *Software and Systems Modeling*, 2007.
53. N. Kececi, J. Garbajosa and P. Bourque, Modeling functional requirements to support traceability analysis, in *2006 IEEE International Symposium on Industrial Electronics*, Vol. 4, 2006, pp. 3305–3310.
54. J. Kelleher, A reusable traceability framework using patterns, in *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering*, New York, NY, USA, 2005, pp. 50–55.
55. B. Kitchenham, Procedures for performing systematic reviews, Technical report, Keele University and NICTA, 2004.
56. M. Lormans and A. van Deursen, Reconstructing requirements coverage views from design and test using traceability recovery via LSI, in *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering*, New York, NY, USA, 2005, pp. 37–42.
57. A. de Lucia, F. Fasano, R. Oliveto and G. Tortora, Recovering traceability links in software artifact management systems using information retrieval methods, *ACM Transactions on Software Engineering and Methodology* **16**(4) (2007) 13.
58. A. Marcus, J. I. Maletic and A. Sergeyev, Recovery of traceability links between software documentation and source code, *International Journal of Software Engineering and Knowledge Engineering* **15**(5) (2005) 811–836.
59. L. Naslavsky, T. A. Alspaugh, D. J. Richardson and H. Ziv, Using scenarios to support traceability, in *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering*, New York, NY, USA, 2005, pp. 25–30.
60. R. P. Noll and M. B. Ribeiro, Enhancing traceability using ontologies, in *Proceedings of the 2007 ACM Symposium on Applied Computing*, New York, NY, USA, 2007, pp. 1496–1497.
61. I. Ozkaya and O. Akin, Tool support for computer-aided requirement traceability in architectural design: The case of DesignTrack, *Automation in Construction* **16** (2007) 674–684.
62. K. Pohl, R. Dömges and M. Jarke, Towards method-driven trace capture, in *Proceedings of the 9th International Conference on Advanced Information Systems Engineering*, 1997, pp. 103–116.

63. B. Ramesh, Factors influencing requirements traceability practice, *Communications of the ACM* **41**(12) (1998) 37−44.
64. B. Ramesh and M. Jarke, Toward reference models for requirements traceability, *IEEE Transactions on Software Engineering* **27**(1) (2001) 58−93.
65. B. Ramesh, C. Stubbs, T. Powers and M. Edwards, Requirements traceability: Theory and practice, *Annals of Software Engineering* **3** (1997) 397−415.
66. R. Ravichandar, J. D. Arthur and M. Perez-Quinones, Pre-requirement specification traceability: Bridging the complexity gap through capabilities, 2007.
67. M. Riebisch and M. Hubner, Traceability-driven model refinement for test case generation, in *Proceedings of the 12th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, Washington, DC, USA, 2005, pp. 113−120.
68. S. Rochimah, W. M. N. Wan Kadir and A. H. Abdullah, An evaluation of traceability approaches to support software evolution, in *Proceedings of the 2007 International Conference on Software Engineering Advances*, Washington, DC, 2007, p. 19.
69. A. M. Salem, Improving software quality through requirements traceability models, in *Proceedings of the IEEE International Conference on Computer Systems and Applications*, Washington, DC, 2006, pp. 1159−1162.
70. C. B. Seaman, Qualitative methods in empirical studies of software engineering, *IEEE Transactions on Software Engineering* **25**(4) (1999) 557−572.
71. Requirements tracing — An overview, http://www.sei.cmu.edu/str/descriptions/reqtracing.html, October 2008.
72. S. A. Sherba and K. M. Anderson, A framework for managing traceability relationships between requirements and architectures, In *Second International Software Requirements to Architectures Workshop Part of 2003 International Conference on Software Engineering*, New York, NY, USA, 2003, pp. 150−156.
73. J. E. Shin, A. G. Sutcliffe and A. Gregoriades, Scenario advisor tool for requirements engineering, *Requirements Engineering* **10**(2) (2005) 132−145.
74. I. Sommerville and G. Kotonya, *Requirements Engineering: Processes and Techniques* (John Wiley & Sons, New York, 1998).
75. I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide* (John Wiley & Sons, New York, 1997).
76. G. Spanoudakis, A. Zisman, E. Perez-Minana and P. Krause, Rule-based generation of requirements traceability relations, *Journal of Systems and Software* **72** (2004) 105−127.
77. D. Streitferdt, Traceability for system families, in *Proceedings of the 23rd International Conference on Software Engineering*, Washington, DC, 2001, pp. 803−804.
78. R. Torkar, T. Gorschek, R. Feldt, U. A. Raja and K. Kamran, Questionnaire with answers, http://dl.dropbox.com/u/2437798/questions_and_answers.pdf, October 2008.
79. R. Torkar, T. Gorschek, R. Feldt, U. A. Raja and K. Kamran, Rejected articles, http://dl.dropbox.com/u/2437798/rej_art.pdf, October 2008.
80. B. Tvete, Introducing efficient requirements management, in *Proceedings of the 10th International Workshop on Database and Expert Systems Applications*, Washington, DC, 1999, p. 370.
81. T. Verhanneman, F. Piessens, B. De Win and W. Joosen, Requirements traceability to support evolution of access control, in *Proceedings of the 2005 Workshop on Software Engineering for Secure Systems — Building Trustworthy Applications*, ACM, New York, USA, 2005, pp. 1−7.

82. C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Wesslén, *Experimentation in Software Engineering: An Introduction* (Kluwer Academic Publishers, Norwell, 2000).
83. S. Yadla, J. H. Hayes and A. Dekhtyar, Tracing requirements to defect reports: An application of information retrieval techniques, *Innovations in Systems and Software Engineering* **1**(2) (2005) 116−124.