Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 112 (2017) 1141–1150

Procedia Computer Science

www.elsevier.com/locate/procedia

21th International Conference on Knowledge Based and Intelligent Information and Engineering Systems

Preliminary Systematic Literature Review of Software and Systems Traceability

Haruhiko Kaiya[a⋆], Ryohei Sato[a], Atsuo Hazeyama[b], Shinpei Ogata[c], Takao Okubo[d], Takafumi Tanaka[e], Nobukazu Yoshioka[f], Hironori Washizaki[g]

[a]*Kanagawa University, Hiratsuka 259-1293, Japan*
[b]*Tokyo Gakugei University, Tokyo 84-8501, Japan*
[c]*Shinshu University, Nagano 380-0928, Japan*
[d]*Institute of Information Security (IISEC), Yokohama 221-0835, Japan*
[e]*Tokyo University of Agriculture and Technology, Tokyo 183-0057, Japan*
[f]*National Institute of Informatics (NII), Tokyo 100-0003, Japan*
[g]*Waseda University, Tokyo 169-8555 , Japan*

Abstract

Traceability is important knowledge for improving the artifacts of software and systems and processes related to them. Even in a single system, various kinds of artifacts exist. Various kinds of processes also exist, and each of them relates to different kinds of artifacts. Traceability over them has thus large diversity. In addition, developers in each process have different types of purposes to improve their artifacts and process. Research results in traceability have to be categorized and analyzed so that such a developer can choose one of them to achieve his/her purposes. In this paper, we report on the results of Systematic Literature Review (SLR) related to software and systems traceability. Our SLR is preliminary one because we only analyzed articles in ACM digital library and IEEE computer society digital library. We found several interesting trends in traceability research. For example, researches related to creating or maintaining traceability are larger than those related to using it or thinking its strategy. Various kinds of traceability purposes are addressed or assumed in many researches, but some researches do not specify purposes. Purposes related to changes and updates are dominant.

Peer-review under responsibility of KES International

Keywords: Systematic Literature Review; Software and Systems Traceability; IEEE CPS; ACM

1. Introduction

The most important artifact in software and systems development is of course a running system including executable programs because the running system directly contributes to the activities of customers. The other artifacts such as requirements specifications, design notes, architecture descriptions and test cases are unwillingly developed

[a] Corresponding author. Tel.: +81-463-59-4111
E-mail address: kaiya@kanagawa-u.ac.jp

1877-0509
Peer-review under responsibility of KES International
10.1016/j.procs.2017.08.152

because the quality of the running system such as completeness (the degree how much the running system provides what the customers require) cannot be guaranteed without such additional artifacts. In addition, the quality of the development process such as efficiency also depends on such additional artifacts. All the artifacts including the running system are related with each other. To guarantee the quality of the artifacts and the process, such relationships among the artifacts should be managed. The researches about software and systems traceability have the fundamental background mentioned above. In a well-known article[1], traceability is defined as follows: "The ability to describe and follow the life of an artifact (requirements, code, tests, models, reports, plans, etc.) developed during the software life cycle in both forward and backward directions $\cdots$". A standard[2] defined traceability as "The degree to which a relationship can be established between two or more products of the development process $\cdots$". Because most running systems today are developed based on existing systems, relationships among artifacts which belong to different processes should be also taken into account.

According to our argument above, we regard that software and systems traceability is very important knowledge. In fact, a plenty of articles has been published in this field. However, methods, techniques and/or tools in such articles have not been applied widely in industry[3]. One of the reasons is that establishing and maintaining traceability links is a time consuming, error prone, and person-power intensive task[4]. We are thus interested in recent research trends in traceability. Systematic Literature Review (SLR)[5] is a useful tool to know such trends. We thus performed SLR on traceability and report the results in this paper. Before performing SLR, we had several bias to traceability researches according to our daily research activities. First, we assumed most researches focused on creating traceability, and they did not focus on its usage enough. Second, we assumed kinds of artifacts were limited although various kinds of artifacts were used in any software processes. Third, we assumed most researches did not provide enough evidences whether their proposals were confident. Research questions (RQs) in this paper are designed based on these pieces of the bias.

Our SLR is a preliminary study because we only focused on articles in ACM digital library and IEEE computer society digital library. These two libraries contains proceedings of the outstanding conferences such as ICSE (International conference of software engineering), ASE (Automated software engineering), and RE (Requirements Engineering). They also contains those of the traceability specific meetings such as SST (Symposium on software and systems traceability) and TEFSE (International workshop on traceability in emerging forms of software engineering). Although the main reasons of this limitation come from our financial and time issues, we can easily guarantee the quality of articles in our review by this limitation.

The rest of this paper is organized as follows: Section 2 describes the details of our review methodology, including research questions (RQs), inclusion and exclusion criteria, our search strategies, and the method for data extraction and analysis. In Section 3, we discussed the results and findings of our review on the basis of our RQs. Threats to the validity are then discussed in Section 4. Finally, we conclude our review and show the future issues.

2. Review Method

In this review, we basically followed the well-known guideline[5] for conducting our SLR. We explain our SLR steps here in detail.

2.1. Research Questions

This review aims at addressing the following Research Questions (RQs):

- **RQ1: What types of researches have been performed in traceability?**
  According to a book[6], traceability researches are categorized into the following types: *Strategy, Creating, Using and Maintaining*.
- **RQ2: Does a traceability research propose *new* idea or *enhance* existing ideas?**
  Because traceability research has its long history[1], there already exists a plenty of techniques and tools. No one obstructs their applications to real world problems. However, problems in this field have not been resolved and new and "novel" researches have been produced almost every year. We thus focus on whether a research is new one or enhanced version of existing ideas.
- **RQ3: What kinds of artifacts do a traceability research cover?**
  We will analyze two aspects of artifacts: representation and contents. Examples of representations are natural language documents, models and action logs. We simply regard diagrams such as directed graphs as models in this paper. By using natural language, we can represent any software engineering artifacts except source codes.

Therefore, we have to distinguish the representation method of an artifact from its contents. We characterize the contents of an artifact according to the development phases: requirements, design, architecture, source codes and tests.

- **RQ4: Does a traceability research clarify *purpose(s)* of its traceability? What kinds of purposes will the research contribute to achieve if the research clarify them?**
  As mentioned in introduction, a traceability research is meaningless if its outcomes are useless in any activities during a software life cycle. In addition, it is too irresponsible to let developers or maintainers to find how to utilize such outcomes such as traceability links. In an article[7], such purposes were partially exemplified: *identifying sources of requirements, impact analysis, test case validation, tracking the progress of a project and identifying interdependencies among artifacts*.
- **RQ5: Does a research show whether a research will contribute to achieve its intended purposes? What kinds of means are used to show that?**
  This research question is related to the evaluation of a research. Examples of evaluation means are as follows: *experiment, argumentation, (formal) proof, case study*. Although experiments and case studies are rigorously defined[8], we simply regard showing examples as a kind of case study. A typical example of the means is an experiment. If the purpose of a traceability research is improving the accuracy of the impact analysis, making a comparative experiment with or without the research outcome is the typical means to show the contribution of the research.
- **RQ6: Are the precision and recall metrics used in the evaluation of the research? How to validate the research by using the metrics if the metrics are used?**
  Especially in an experiment or a case study, the precision and the recall are the typical metrics to validate the research outcome. To calculate these metrics, *the correct answers (oracle)* are required. The correct answers depend on the purpose of the traceability research. For example, actually impacted artifacts should be identified in advance when researchers expect their research is useful for impact analysis. The precision corresponds to the accuracy of predicted results by a research, and the recall corresponds to its completeness, i.e. the degree without omission.

2.2. Inclusion and Exclusion Criteria

We use the following inclusion and exclusion criteria for our SLR.

- Articles in ACM digital library and IEEE computer society digital library were included.
- Articles published in June 2006 to September 2016 were included, and other articles were excluded. Note that this research was started at the end of September 2016.
- Traceability related to software and software intensive systems was included, and others were excluded.
- Articles in magazines such as CACM or IEEE computer were excluded.
- If the same article contains in both libraries, one of them were excluded.

2.3. Search Strategy

To retrieve articles as many as possible, we used single word “traceability” for the search page of each library. The authors then examined the abstract manually to exclude articles according to the criteria above.

2.4. Data Extraction and Synthesis Method

We follow the steps below for extracting data and synthesize them.

1. Articles to be reviewed are downloaded and shared by authors.
2. An author prepare a form for analyzing each article. The form contains 11 questions and they will be briefly explained below.
3. For each article, two authors fill the form respectively by reading the contents of the article.
4. Another author examines two filled forms for each article. If there are some inconsistencies in these forms, the author examines the contents of the article, and fixes the inconsistencies.
5. All the results are arranged and sorted for answering the RQ’s.

The form contains following 11 questions.

Table 1. Reviewed articles

Source	IEEE CSDL	ACM	Total
Number of articles	36	23	59

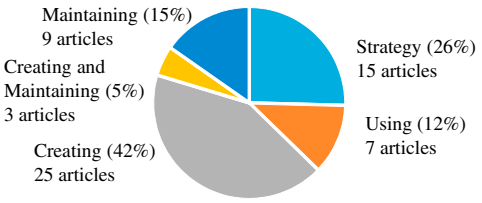

Fig. 1. RQ1: ratio of research types

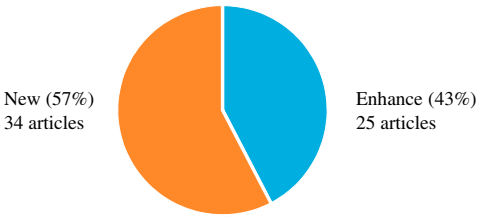

Fig. 2. RQ2: ratio whether a research is new or enhancement

- Q1 for RQ1: What is the research type of this article? (Choose one of them: strategy, using, creating, maintaining)
- Q2 for RQ2: Is the research in this article new or enhanced version of existing research? (Choose one: new, enhance)
- Q3 for RQ3: What kinds of artifacts are dealt with in this articles? (Fill the list of artifacts such as Java, UML, and DFD)
- Q4 for RQ4: Is the purpose of the traceability clarified in this article? (Yes or No)
- Q5 for RQ4: What are the purpose(s) of the traceability in this article? (Fill the list of purposes such as impact analysis, test case validation, dependency identification and so on)
- Q6 for RQ5: Is the means for confirming whether the purposes are achieved shown in this article? (Yes or No)
- Q7 for RQ5: What kinds of means are used in this article? (Fill the means such as case study, experiment or proof)
- Q8 for RQ6: Is precision used for validation in this article? (Yes or No)
- Q9 for RQ6: What kind of criteria is used to decide the precision is good or bad? (Fill the criteria)
- Q10 for RQ6: Is recall used for validation in this article? (Yes or No)
- Q11 for RQ6: What kind of criteria is used to decide the recall is good or bad? (Fill the criteria)

3. Results and Findings

Articles to be reviewed are summarized in Table 1, and concrete lists of the articles are attached to the end of this paper. As a result, most articles are conference, workshop or symposia papers.

3.1. RQ1: Research Type

We categorized articles into the following four types:

- Strategy: A research investigating what kinds of traceability are needed, and planning how to implement the traceability is categorized into strategy.
- Using: A research related to the usage of traceability is categorized into using.
- Creating: A research proposing how to create or to recover traceability links is categorized into creating.
- Maintaining: A research how to update the existing traceability links is categorized into maintaining.

The result is shown in Figure 1. Because Creating and Maintaining are a little bit similar, three articles addressed these two types. As shown in the result, the types of Creating or Maintaining take large portion of all articles, i.e. more than 60 %. Applying natural languages processing techniques and statistical techniques was dominant in such types because such techniques were easy to apply artifacts to be traced.

Because creating and maintaining traceability is just a means, how to using traceability or what it should be are discussed together with creating and maintaining it. This issue is analyzed in RQ4.

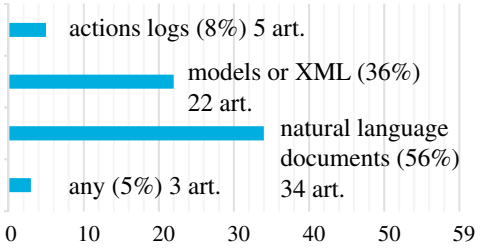

Fig. 3. RQ3: portions of each representation types

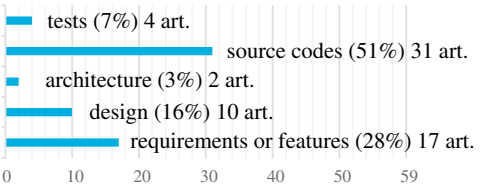

Fig. 4. RQ3: portions of each contents types

3.2. RQ2: New or Enhance

At least more than 20 years, traceability has been studied [1]. Articles based on existing researches may thus exist. On the other hand, new and novel ideas should be proposed because environments and domains for software and systems have been changed every day. Figure 2 shows the portion whether a research is new or enhancement of existing one. As shown in the figure, half of the articles is new, and other half is enhancement. The ratio of this portion seems very reasonable because both types of researches are performed simultaneously.

3.3. RQ3: Covered artifacts

We identified following four types of representation in reviewed articles.

- any: any kinds of representation.
- natural language documents: Requirements specifications or design notes are normally written in natural language such as English.
- models or XML: structural and graphical notations such as UML are frequently used in software and systems development. Although XML has not its own graphical notation, XML is categorized into this kind due to its structural feature.
- actions logs: bug tracking systems can record some action logs, and such logs can be referred. Even verbal data in a meeting can be stored and referred. Some traceability techniques can deal with such pieces of artifacts.

Figures 3 shows portions of each representation. Because a research in an article deals with more than two representation types, we do not use a pie chart. As shown in the figure, more than half articles deal with natural language documents. It is reasonable because using words and phrases in such documents is useful to create traceability links. In addition, most requirements are written in natural languages. Structural representations such as models or XML are also dealt by more than one third articles. It is also reasonable because it is easy to create traceability links over such representations.

We also identified following five contents types in reviewed articles.

- requirements or features: We regard features in software product lines are kinds of requirements.
- design
- architecture
- source codes
- tests: test cases or test documents.

Figure 4 shows portions of each content. Because a research in an article deals with more than two contents types in the same way as the representations types, we do not use a pie chart. As shown in the figure, source codes are dealt with by about half of articles. It seems to be reasonable because source codes are structured and their components such as classes, methods and functions can be easily identified. About one fourth articles deal with requirements or features. Because requirements are the base points in traceability, its portion should be larger than the portion shown in the figure.

Finally, we identified concrete artifacts dealt in reviewed articles. Java and UML are frequently dealt, 14 and 7 articles respectively. Except Java and UML, C, GSN, AspectJ and AOP are dealt. Because Java and UML are the most popular artifacts in the current development, this result seems to be reasonable.

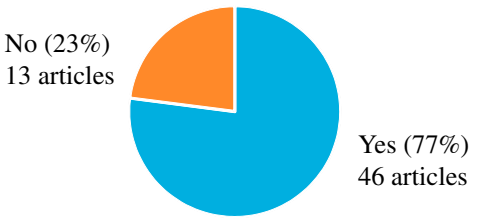

Fig. 5. RQ4: ratio whether traceability in an article clarifies its purpose or not

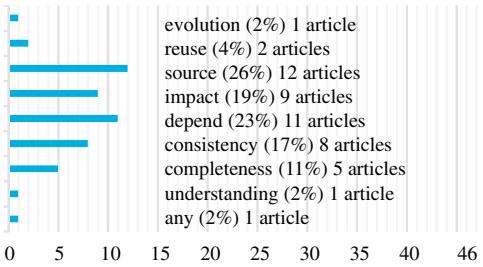

Fig. 6. RQ4: portions of purpose types

3.4. RQ4: Purposes

As mentioned in introduction, the purpose of traceability is really important because traceability itself is a secondary or thirdly artifact. A main artifact in software and systems development is the running system. Figures 5 shows the ratio whether traceability in an article clarifies its purpose or not. As shown in the figure, the purpose is clarified in most articles, but about one fourth articles do not. In some articles, creating links becomes the purpose of their research, and their usages and purposes are rarely discussed. Even when creating links is the main issue of a research, its usages and purposes should be stated or discussed.

We identified nine types of purposes of traceability usage.

- any: In an article, the authors state its traceability can use any purposes.
- understanding: The traceability contributes to improving the understanding of developers or maintainers.
- completeness: The traceability contributes to identifying the missing artifacts. For example, links between requirements and codes contribute to finding unimplemented requirements.
- consistency: The traceability contributes to finding inconsistency among artifacts. For example, incorrectly implemented requirements can be found with the help of links between requirements and codes.
- depend: The traceability contributes to knowing the dependencies among artifacts.
- impact: The impact analysis is the typical purpose of traceability usage.
- source: The traceability contributes to identify the origin of an artifact. Note that this type is not specific purpose related to source codes.
- reuse: The traceability contributes to reusing a part of existing artifacts.
- evolution: The traceability contributes to updating existing software and systems.

The types “evolution”, “source”, “impact”, “depend” and “reuse” are related to changing or updating existing software and systems. On the other hand, the types “consistency”, “completeness” and “understanding” are related to analyzing current software and systems.

Figure 6 shows the portions of purpose types. Because traceability in an article contributes to more than two purposes, we do not use pie chart. As shown in the figure, there is no outstanding type. Three types related to changing and updating software and systems are the most dominant purposes. The result seems to be reasonable because changes and updates of artifacts are the most important concerns in the current development due to the frequently and suddenly requirements changes.

3.5. RQ5: Means for confirming purpose achievement

As shown in Figure 5, 46 articles specify the purpose of their traceability. We thus analyze how to confirm whether the purpose is achieved in each article, i.e. the evaluation. Figure 7 shows the ratio whether the authors of an article provide the means for confirming that the purpose of their traceability is achieved. As shown in the figure, such means are provided by more than 80% articles. This result is reasonable because articles are rarely accepted today without evaluation.

We then focus on the types of evaluation. Figure 8 shows the ratio of evaluation means types. A means “case” is the most dominant, and the second dominant is “exp”. The other types are very few. Although rigorous case study is hard to be performed[8], we regard showing an example as “case”. That is one of the reason the type “case” is dominant. In this sense, the evaluation means is a little bit weak in more than 60% articles.

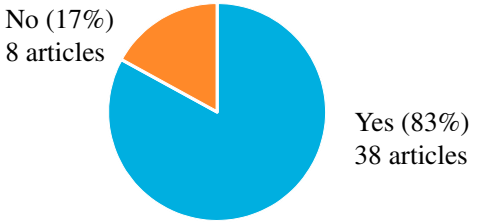

Fig. 7. RQ5: ratio whether the purpose achievement of an article is confirmed or not

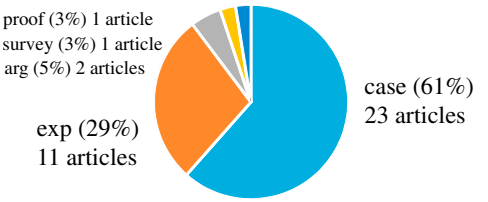

Fig. 8. RQ5: ratio of evaluation means types

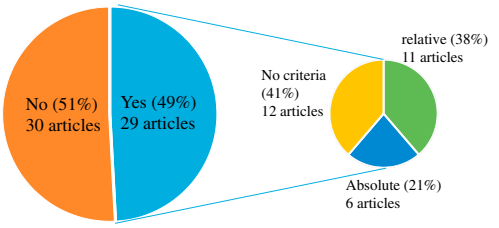

Fig. 9. RQ6: rate of precision usage and rate of its evaluation criteria

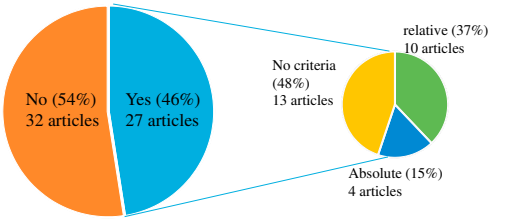

Fig. 10. RQ6: rate of recall usage and rate of its evaluation criteria

3.6. RQ6: Precision and Recall

Precision and recall are the typical metrics especially for evaluating whether predicted links are correct and complete. In both cases, correct traceability is required. Precision shows the degree whether predicted links are correct. In this case, researchers may simply check each predicted link step by step. Recall shows the degree whether no missing correct links exist by prediction. In this case, correct traceability is required in advance, and all parts of all artifacts should be examined for identifying correct traceability in general. Therefore, using recall is more difficult than using precision in general.

Even if the purpose of the traceability in an article is not specified (see Figure 5), precision or recall can be used in the article. We thus analyze all 59 articles in RQ6. We categorize evaluation criteria of precision or recall into the following three types.

- relative: Precision or recall values of the technique in an article is compared to the value of existing techniques. When the trade-offs between precision and recall values are analyzed, we also regard the article has the relative criterion.
- absolute: The authors give some absolute criterion such as 80% or 100% for their evaluation.
- No criteria: there are no criteria, and just the values are shown and discussed.

Left pie chart in Figure 9 shows whether precision is used in an evaluation of an article. About half of articles use precision for their evaluation. As shown in the right pie chart in the figure, the rate of evaluation criteria is shown. More than 40% of articles do not give criterion for precision evaluation. According to the result, the quality of evaluation is not high.

Figure 10 shows the results of recall in the same way as the results of precision in Figure 9. The trend is almost the same as the results of precision. As mentioned at the beginning of this section, using recall is more difficult than using precision. However, the results show recall is used in the same rate as precision. In many purposes of traceability, recall should be complete, i.e. 100% because missing links seriously affect the quality of development activity. For example, all necessary modifications are not performed by using an impact analysis technique without 100% recall. Even if the precision is not 100%, a developer performing impact analysis may simply skip the incorrect links. In this sense, the portion of absolute criterion should be larger than the portion of a pie chart at the right side in Figure 10.

4. Threats to Validity

The most serious threat to the validity of this research is the comprehensiveness of reviewed articles, i.e. the completeness of research range. As mentioned in Section 2.2, we only focus on the last 10 year's articles in ACM digital library and IEEE computer society digital library. We understand articles in high quality exist outside these two libraries such as LNCS of springer. In addition, most articles we analyzed are conference papers. To mitigate this threat, we will use commercial bibliographic databases such as Scopus in our next step. On the other hand, the quality of articles can be preserved by narrowing down the range of libraries. Basically, both ACM and IEEE CS libraries contains articles published in highly competitive conferences such as ICSE, ASE or RE. Although this fact never mitigates the threat about comprehensiveness, it contribute to improving the overall quality of our SLR.

The second threat is about the comprehensiveness of our research framework. Our framework consists of several viewpoints and each of them basically corresponds to each research question. Our research questions are based on our previous bias mentioned in Intoroduction, and they largely influenced by a book[6] and an article[7]. For example, research types in RQ1 comes from the book, and purposes in RQ4 comes from the article. Some missing viewpoints can thus exist in our analysis framework for software and systems traceability. In addition, some existing viewpoint should be elaborated. For example, we simply enumerate the purpose types in RQ4. However, some dependency structure seems to be required in purpose types because some purposes contributes to another purpose, i.e. sub-purposes of the purpose. We may use goal-oriented modelling notations for this structuring.

5. Conclusion

Traceability methods and techniques have been continuously proposed and evaluated in literatures. By conducting our SLR about software and systems traceability, we gathered, classified and analyzed such techniques from research questions (RQs) in Section 2.1. The RQs are based on our previous bias mention in Introduction. We reviewed 59 articles from two well-known digital libraries. Although the resources of our review are limited, the quality of these papers were not bad. We have the following five findings of this review. First, more researches about using traceability or thinking its strategy are needed to enable a user of traceability techniques to choose some of them suitable for him/her. Second, various kinds of artifacts should be dealt for each technique. Third, the usage of each traceability technique should be definitely clarified although it was not clear in some articles. Fourth, traceability purposes related to consistency and completeness may be encouraged more than now because purposes of changing and updating software and systems have been investigated more than the purposes related to consistency and completeness. Fifth, rigorous evaluation such as a comparative experiment should be encouraged. We clarified our future issues in Section 4 as threats to the validity: comprehensiveness of reviewed articles and the research framework.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 16H02804, 15K00109 and 15H02686.

References

1. Gotel, O.C.Z., Finkelstein, A.. An analysis of the requirements traceability problem. In: *Proceedings of the First IEEE International Conference on Requirements Engineering, ICRE '94, Colorado Springs, Colorado, USA, April 18-21, 1994*. 1994, p. 94–101.
2. IEEE, . IEEE Standard Glossary of Software Engineering Terminology. 1990. IEEE Std 610.12-1990.
3. Kaiya, H., Hara, K., Kobayashi, K., Osada, A., Kaijiri, K.. Exploring how to support software revision in software non-intensive projects using existing techniques. In: *Workshop Proceedings of the 35th Annual IEEE International Computer Software and Applications Conference, COMPSAC Workshops 2011, Munich, Germany, 18-22 July 2011*. 2011, p. 327–334.
4. Ramesh, B., Jarke, M.. Toward reference models of requirements traceability. *IEEE Trans Software Eng* 2001;**27**(1):58–93.
5. Kitchenham, B.. Procedures for performing systematic reviews. Tech. Rep. 0400011T.1; NICTA; The address of the publisher; 2004.
6. Cleland-Huang, J., Gotel, O., Zisman, A.. *Software and Systems Traceability*. Springer Publishing Company, Incorporated; 2012. ISBN 1447122380, 9781447122388.
7. Torkar, R., Gorschek, T., Feldt, R., Svahnberg, M., Raja, U.A., Kamran, K.. Requirements traceability: a systematic review and industry case study. *International Journal of Software Engineering and Knowledge Engineering* 2012;**22**(3):385–434.
8. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.. *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers; 2000. ISBN 0-7923-8682-5.

List of reviewed papers from IEEE CSDL

1. C. Chang, H. Jaygarl, I.-X. Chen, T. Nguyen, and H.-Y. Jiang, "Incremental latent semantic indexing for automatic traceability link evolution management," *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, pp. 59–68, 2008.
2. A. Lozano, C. Noguera, and V. Jonckers, "Managing traceability links with matraca," *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 01, pp. 665–668, 2016.

3. A. Abadi, M. Nisenson, and Y. Simionovici, "A traceability technique for specifications," *International Conference on Program Comprehension*, pp. 103–112, 2008.
4. P. Mader, O. Gotel, T. Kuschke, and I. Philippow, "tracemaintainer - automated traceability maintenance," *2008 16th IEEE International Requirements Engineering Conference*, pp. 329–330, 2008.
5. R. Pooley and C. Warren, "Reuse through requirements traceability," *Software Engineering Advances, International Conference on*, pp. 65–70, 2008.
6. S. Kang, J. Kim, S. Kang, and S. Eom, "A formal representation of platform feature-to-requirement traceability for software product line development," *2014 IEEE 38th Annual Computer Software and Applications Conference (COMPSAC)*, pp. 211–218, 2014.
7. B. V. Rompaey and S. Demeyer, "Establishing traceability links between unit test cases and units under test," *2011 15th European Conference on Software Maintenance and Reengineering*, pp. 209–218, 2009.
8. H. Schwarz, "Towards a comprehensive traceability approach in the context of software maintenance," *2011 15th European Conference on Software Maintenance and Reengineering*, pp. 339–342, 2009.
9. J. Helming, J. David, M. Koegel, and H. Naughton, "Traceability rearmed," *2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2009)*, vol. 01, pp. 340–348, 2009.
10. R. Oliveto, A. Panichella, A. D. Lucia, S. Panichella, and G. Capobianco, "Traceability recovery using numerical analysis," *2009 16th Working Conference on Reverse Engineering. WCRE 2009*, pp. 195–204, 2009.
11. S. Hayashi, T. Yoshikawa, and M. Saeki, "Sentence-to-code traceability recovery with domain ontologies," *2010 17th Asia Pacific Software Engineering Conference (APSEC 2010). Software for Improving Quality of Life*, pp. 385–394, 2010.
12. Y.-G. Gueheneuc, G. Antoniol, and N. Ali, "Trust-based requirements traceability," *International Conference on Program Comprehension*, pp. 111–120, 2011.
13. T. Kobayashi, Y. Ichikawa, and S. Nagano, "Recovering traceability links between code and documentation for enterprise project artifacts," *2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC 2012)*, pp. 11–18, 2012.
14. A. Panichella, A. D. Lucia, and A. Zaidman, "Adaptive user feedback for ir-based traceability recovery," *2015 IEEE/ACM 8th International Symposium on Software and Systems Traceability (SST)*, pp. 15–21, 2015.
15. T. Beyhl and H. Giese, "Traceability recovery for innovation processes," *2015 IEEE/ACM 8th International Symposium on Software and Systems Traceability (SST)*, pp. 22–28, 2015.
16. S. Namdar and M. Mirakhorli, "Toward actionable software architecture traceability," *2015 IEEE/ACM 8th International Symposium on Software and Systems Traceability (SST)*, pp. 36–42, 2015.
17. K. Taguchi, S. Daisuke, H. Nishihara, and T. Takai, "Linking traceability with gsn," *2014 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pp. 192–197, 2014.
18. J. Jurjens, J. Schreck, and Y. Yu, "Tools for traceability in secure software development," *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, pp. 503–504, 2008.
19. X. Chen and J. Grundy, "Improving automated documentation to code traceability by combining retrieval techniques," *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, pp. 223–232, 2011.
20. G. Bavota, L. Colangelo, A. D. Lucia, S. Fusco, R. Oliveto, and A. Panichella, "Traceme: Traceability management in eclipse," *2013 IEEE International Conference on Software Maintenance*, pp. 642–645, 2012.
21. Y. Laghouaouta, A. Anwar, and M. Nassar, "A traceability approach for model composition," *2013 ACS International Conference on Computer Systems and Applications (AICCSA)*, pp. 1–4, 2013.
22. H. Eyal-Salman, A.-D. Seriai, and C. Dony, "Feature-to-code traceability in legacy software variants," *2013 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 57–61, 2013.
23. N. Alhindawi, O. Meqdadi, B. Bartman, and J. I. Maletic, "A tracelab-based solution for identifying traceability links using lsi," *2013 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)*, pp. 79–82, 2013.
24. G. Bavota, A. D. Lucia, R. Oliveto, A. Panichella, F. Ricci, and G. Tortora, "The role of artefact corpus in lsi-based traceability recovery," *2013 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)*, pp. 83–89, 2013.
25. A. Mahmoud and N. Niu, "Supporting requirements traceability through refactoring," *2013 21st IEEE International Requirements Engineering Conference (RE)*, pp. 32–41, 2013.
26. J. M. Vara, V. A. Bollati, Á. Jiménez, and E. Marcos, "Dealing with traceability in the mddof model transformations," *IEEE Trans. Software Eng.*, vol. 40, no. 6, pp. 555–583, 2014. [Online]. Available: `http://dx.doi.org/10.1109/TSE.2014.2316132`
27. O. Badreddin, A. Sturm, and T. C. Lethbridge, "Requirement traceability: A model-based approach," *2014 IEEE 4th International Model-Driven Requirements Engineering Workshop (MoDRE)*, pp. 87–91, 2014.
28. D. Ai, N. Ubayashi, P. Li, S. Hosoai, and Y. Kamei, "iarch - an ide for supporting abstraction-aware design traceability," *2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pp. 442–447, 2014.
29. P. Rempel and P. Mader, "A quality model for the systematic assessment of requirements traceability," *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pp. 176–185, 2015.
30. J. H. Hayes, G. Antoniol, B. Adams, and Y.-G. Gueheneuc, "Inherent characteristics of traceability artifacts less is more," *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pp. 196–201, 2015.
31. N. Mustafa and Y. Labiche, "Towards traceability modeling for the engineering of heterogeneous systems," *2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pp. 321–328, 2015.
32. A. Noyer, P. Iyenghar, E. Pulvermueller, F. Pramme, and G. Bikker, "Traceability and interfacing between requirements engineering and uml domains using the standardized reqif format," *2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pp. 1–6, 2015.
33. K. Nishikawa, H. Washizaki, Y. Fukazawa, K. Oshima, and R. Mibe, "Recovering transitive traceability links among software artifacts," *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 576–580, 2015.
34. G. Tortora, A. D. Lucia, F. Fasano, and R. Oliveto, "Can information retrieval techniques effectively support traceability link recovery?" *14th IEEE International Conference on Program Comprehension*, pp. 307–316, 2006.

35. F. Dong, C. K. Chang, H. yi Jiang, and T. N. Nguyen, "Traceability link evolution management with incremental latent semantic indexing," *2007 31st Annual International Computer Software and Applications Conference. COMPSAC 2007*, vol. 01, pp. 309–316, 2007.
36. A. H. Abdullah, S. Rochimah, and W. M. N. W. Kadir, "An evaluation of traceability approaches to support software evolution," *2007 International Conference on Software Engineering Advances*, p. 19, 2007.