Journal of Engineering and Applied Sciences 12 (Special Issue 4): 6870-6877, 2017
ISSN: 1816-949X

A Systematic Literature Review of Traceability Practices for Managing Software Requirements

[1]Mazni Omar and [1, 2]Jasim Mohammed Dahr
[1]Human-Centered Computing Research Lab., School of Computing,
Universiti Utara Malaysia, Changlun, Malaysia
[2]Directorate of Education, Basrah, Iraq

Abstract: Requirement Traceability (RT) is one of the crucial activities in managing requirements. In addition, RT is important part of developing software projects. Research has shown that RT influences positively on the quality of software products. Nevertheless, there is a lack of guidance available for practitioners to assist them to establish effective traceability in their projects. In this study, we present Systematic Literature Review (SLR) of requirement traceability practices. This study enables practitioners to find a suitable method for tracing requirements and also enables researchers to find gaps and pointers for future study in this domain.

Key words: Requirement traceability practices, systematic literature review, guidance, traceability, developing, influences

INTRODUCTION

Requirements Traceability (RT) is defined as "the ability to describe and follow the life of a requirement in both a forward and backward direction, e.g., from its origins, through its development and specification, to its subsequent deployment and use and through periods of ongoing refinement and iteration in any of these phases (Gotel and Finkelstein, 1994). RT is one of the crucial activities of good requirements management and allows quick assessment of the impact when a change occurs (Ooi *et al*., 2014). RT is an important part of development projects (Mader and Egyed, 2014). Therefore, researchers have shown that RT influences positively on the quality of software products (Cleland-Huang *et al*., 2014; Winkler and Pilgrim, 2010). In addition, the direct benefits of traceability are an improved product quality, controlled requirements, effective response to changes (Kirova *et al*., 2008). In the same vein, numerous studies also pointed out that the use of requirement traceability in a development project is often hindered by problems in its implementation and application such as, insufficient tool support and ad-hoc traceability without strategy (Reshef *et al*., 2006; Schwarz *et al*., 2010).

RT is considered as the heart of requirement management process. Despite the fact that RT is a topic of much interest in development projects (Rempel *et al*., 2013), surprisingly, it is rarely used (Ahmad and Ghazali, 2007; Klimpke and Hildenbrand, 2009). Furthermore, most of companies are either not implementing it or they are lacking of guidance on how to implement effective traceability (Regan *et al*., 2012). In addition to the aforementioned arguments, there is a lack of guidance is available for practitioners to assist them establish effective traceability in their projects (Casey and Mc Caffery, 2011). As a result of this, practitioners are ill-informed as to how best to accomplish this task such as found in software companies. This opinion is supported by Regan *et al*. (2014).

Notwithstanding the lack of guidance, there are a number of commonly accepted practices which can direct practitioners regarding the implementation and maintenance of traceability (Han *et al*., 2014). In the same context, software companie's requirements are the greatest challenge to handle due to constant modification (Rosmadi *et al*., 2015). Therefore, to aid software companies in addressing the lack of guidance on how to implement effective traceability, this study aims at examining the practices for traceability to be adopted in software companies.

Literature review: The research by Ramesh and Jarke (2001) was on a large practitioner study of traceability which took over 3 years for data collection in the 1990's. The 58 students of masters in information technology were included in a pilot study to produce an opening traceability meta-model and document the scheme and achievements of the work. The 30 focus group

Corresponding Author: Mazni Omar, Human-Centered Computing Research Lab., School of Computing, Universiti Utara Malaysia, Changlun, Malaysia

discussions consisting of 5 people (each from 26 companies) made up the main study. Their main emphasis was the kind of traceability link that was utilized in ideal and current practices.

According on the way to Ahmad and Ghazali (2007) study, their study was conducted on 15 practitioners and 3 IT companies. The practitioners who develop small projects with practical experience of 6-10 years were interviewed. Also, the project documentation of the companies was analyzed. The result of their findings is that subjects viewed pre-requirements traceability as being more useful than post-requirements traceability. The researchers failed to expose the importance or advantages of traceability to their topics in details.

MATERIALS AND METHODS

SLR is a method of making sense of large bodies of information and a means of contributing to the answers to questions about what works and what does not and also to many types of questions (Petticrew and Roberts, 2008). According to many of the researchers who asserted that, SLR is one secondary study method that has gotten much interest lately in many disciplines (Dyba *et al*., 2005; Hannay *et al*., 2007). In addition, SLR should always have a protocol (O'Connor *et al*., 2014). Thus, for conduct the SLR, this study adopted a specific protocol that was carried out based on the steps defined by Kitchenham and Charters (2007). Figure 1 depicts the major steps of systematic review.

Formulating the SLR questions: The systematic literature review aims to answer the research question and find the techniques used in RT which in some of the studies referred to by the term "practices". Systematic review always has one or more questions to answer it and the first step is to determine the research question in any review. The question of SLR is "What are the requirement traceability practices that have been used within software companies?"

Constructing the search: SLR search strategy is a step aimed at providing a base for the unbiased and comprehensive gathering of research from the previous studies. This step necessitates a carefully planned search strategy to ensure that every significant bibliographic citation possess a great chance of appearing in search results. Figure 2 shows the period of SLR, digital libraries and keywords (search terms) used in the review.

Study selection: In fact, not all of the studies will be included, there is a set of standards that will be on the

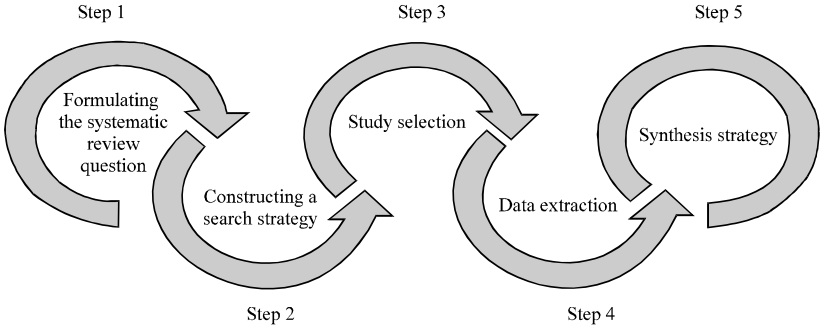

Fig. 1: A systematic review guideline process

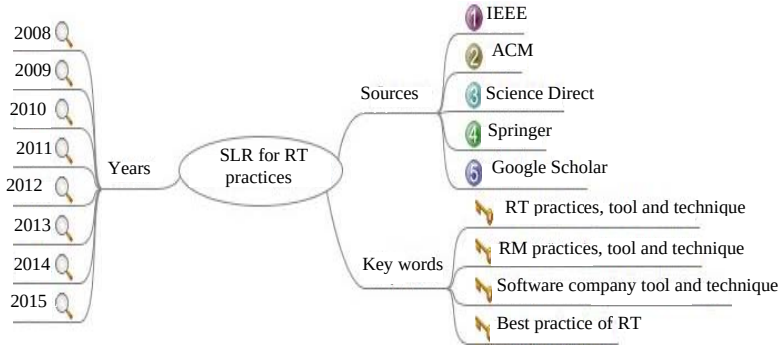

Fig. 2: The search strategy of the SLR

basis of which the exclusion and the inclusion of studies. Exclusion and inclusion criteria are utilized to decide the appropriateness of publications (articles) and to come up with decisions as to whether to reject or accept a specific publication into the SLR. The exclusion conditions or criteria comprises of etudies that lack significance to the research question, studies that is not within the specified period, written studies languages other than english, refined studies in more than a digital library and in addition to studies that do not contain techniques or tools. Whilst the inclusion criteria consist of studies that are contains empirical study, studies within the study period, studies that contain important information about the techniques and traceability tools, studies that are more than 2 page and in addition to studies that are relevant to the research question.

Furthermore, in this study, we concentrate on studies published during the period from 2008 and 2015 which included a techniques and tools used in RT. The time span from 2008-2015 for SLR of this study is deemed act as a complement study for the prior literature, where the latest study was in 2007 while there are important new techniques and tools for tracing requirements.

Data extraction: This step aims to design the forms of data extraction in order to accurately record the researcher information acquired by through the initial studies. Data extraction also involves drawing up a detailed table describing every study (not all studies that were located in the review, only those studies that meet all the inclusion criteria) (Petticrew and Roberts, 2008). Data extraction strategy set by Salvador *et al.* (2014) will adopt in this study.

Synthesis of the extracted data: The last step from the systematic reviews strategies is called "Synthesis". This step involves collating and summarizing the results of the included primary studies. To generate the first pool of articles, citation search was done with the exclusion of additional constraints. Identification of the overall 209 citations has been realized in this initial step via the search strategy. Subsequently, each of the citations was reviewed by the researcher and a set of citations that could be significant by reading the abstracts, introductions and conclusions were selected. The result from this round was 95 studies. More specifically, after reading all 95 studies to identify practices of RT, the inclusion criteria (for this stage) were met by 37 of 92 studies. Moreover, the selected studies were reviewed by two experts of the research methodology and management information system. The details concerning the amount of selected studies discovered

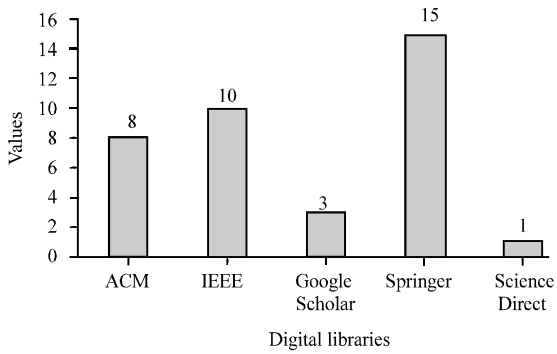

Fig. 3: The selected studies details of digital libraries

while conducting the search are presented on Fig. 3 and the selected studies can be downloaded through this link: https://www.dropbox.com/s/6fijujnhiul0xbs/SLR%20selected% 20 Stuties.pdf?dl = 0

RESULTS AND DISCUSSION

To understand what are the practices used to trace requirements during software development life cycle, SLR questions were established to identify the techniques and tools used in RT. The research questions were answered by examining the literature on RT. Based on the results from the SLR process; the following techniques were used with requirement traceability according to included criteria: Rule-Based (RB), Value-Based (VB), Information Retrieval (IR), Scenario-Based (SB), Event-Based (EB), Hypertext-Based (HTB), ArchTrace, Model-Based (MB) and Goal-Centric (GC) as depicted in Fig. 4.

With regards to RT techniques, the majority of the studies were about IR technique which included 15 studies. The 7 studies contained information about the rules-based technique. As for the other techniques, information pertaining to them were taken from studies within the range of 2-4 studies. In addition, the researcher found the following traceability tools during the review of the literature: Requirement Traceability Matrix (RTM), RETRO, traceMAINTAINER, DesignTrack and DevComplete as depicted in Fig. 5.

Figure 5 shows the number of studies that contained information about requirements traceability tools. Then 6 studies included information on traceMAINTAINER tool. While 4 studies included information on RETRO, 3 studies contained details on RTM tool. Finally, only one study covered Design Track and a similar result was obtained for DevComplete as well.

RB is a technique presented by Spanoudakis *et al.* (2004). RB is one of the wide-spread techniques for maintenance of post-requirements traceability relations.

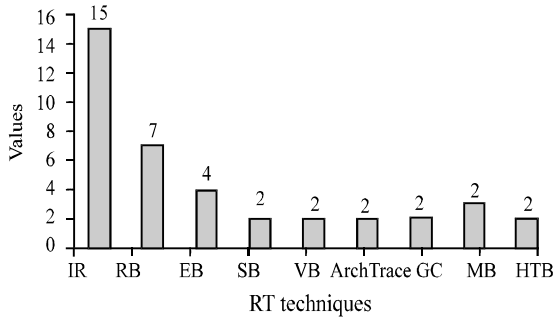

Fig. 4: RT techniques for tracing requirements

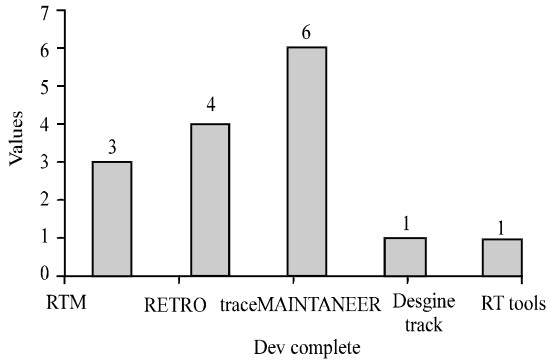

Fig. 5: RT Tools for tracing requirements

RB is a technique which anchors the automatic maintenance of traceability links between analysis, requirements and design models of software systems. VB is a technique proposed by Heindl and Biffl (2005). This technique provides an economic model and a technical model for requirements tracing. It is a measuring the worth that requirement traceability can deliver to a company or an organization for the purpose of supporting decisions associated to the implementation of requirement traceability. This technique can determine requirements that are very important to trace and also requirements that are less valuable to trace.

Many researchers have tried to use IR technique to establish traceability link between different software artefacts (Gotel and Finkelstein, 1994; Egyed and Grunbacher, 2002). IR technique concentrates on automating the creation of traceability link using comparison based on similarity between 2 types of artifacts. IR has been implemented on different tools with RETRO as an example of such tools. The scope of tracing covers almost all artifacts including high-level and low-level requirements, design elements, manual documents, source code and test cases. Egyed and Grunbacher (2002) proposed a SB approach. SB uses assumed trace information that has to be entered manually. After this, it utilizes runtime information to generate trace links. SB has three pre-requisites in order to create traceability link between artefacts. Firstly, there must be an executable or simulate-able software system which can be either partial implementation or incremental prototype. Secondly, a software model must be available for the system. Finally, there must be executed test cases or scenarios.

EB is a technique proposed by Huang *et al.* (2002) for managing and maintaining traceability between requirements and UML along with test cases. EB is a semi-automated procedure for gathering Requirement Traceability (RT) links for relating developers and code artefacts to use cases. The production of trace links was done by monitoring events started by a developer operating in the context of a use case, on a code artefact.

The HTB technique is proposed by Maletic *et al.* (2003) for traceability between different artefacts. HTB is a technique used for traceability link generation using open hypermedia and information integration. This technique is semi-automated, hence, user input is necessary for different tasks (documents, selecting source code and determine threshold value for traceability) during the HTB process.

Murta *et al.* (2008) presented the ArchTrace technique for the advancement of traceability connections between the architecture and implementation. ArchTrace differentiates itself by frequently updating traceability relations or links from architectural elements to code elements via. a policy-based extensible infrastructure which permits developers or a group of developers to choose a set of traceability management policies that is suitable for their situational needs and working styles.

MB is a technique proposed by Huang *et al.* (2009) for RT. The purpose of the MB technique is to determine artifact granularity suitable for traceability and to extract all candidate traceability links in order to achieve a recall rate of 100%. MB technique are developed to assist organizations obtain complete advantage from the traces constructed and to permit the project stakeholders to plan and produce trace schemes and to execute them in a modeling environment which is in a graphical form.

GC is an approach proposed by Huang *et al.* (2005) for trace Non-Functional Requirements (NFRs). It provides traceability anchorage for maintaining and managing (NFRs) and associated quality concerns through the long life span of a software intensive system. The advantages of this technique are that they deliver greater degrees of automation for utilizing and comprehending traceability links and in certain cases are developed precisely with maintainability in mind.

In addition, this study also concentrated on a set of traceability tools. One of these tools is requirement traceability matrix. RTM are mostly utilized in industries to determine the relationships between requirements and other kinds of artifacts such as test cases, code modules and design (Cuddeback *et al*., 2010). More specifically, it is a document that links requirements with the conforming test cases. Each of the requirements is itemized in a matrix row and the matrix columns are used to locate where and how each of the requirements has been addressed. It is also utilized in the management of modifications and as the base for test planning. A good traceability matrix also provides ease in track changes. Hence, the quality of a system is improved (Athira and Samuel, 2011).

Requirements tracing by manual traceability methods are prone to errors and also time-consuming. Therefore, Requirements Tracing on-target (RETRO) is an RT tool presented by Hayes *et al*. (2007), to ease the automatic creation of RTM. Result revealed that RETRO discovered considerably more accurate links than manual tracing (RTM) and used only one third of the usual time to do so (Shahid *et al*., 2011).

traceMAINTAINER is a prototype tool which anchors the semi-automated update of traceability connections between analysis, requirements and design models of the software systems which are expressed in UML (Mader *et al*., 2009). It is a tool that supports a technique for preserving post-requirements traceability connections or relations after modifications have been applied to the traced model elements. traceMAINTAINER allows the update of traceability relations by developers, with little manual effort (Mader *et al*., 2008).

DesignTrack as a traceability tool is a prototype tool which provides an incorporated design environment for form exploration and requirement specification in one design session (Ozkaya and Akin, 2007). It offers an environment for the navigation of intricate design information spaces through supporting requirement traceability. It also calculates the power, applicability and limitations of computer-aided designs which are enabled by requirement traceability. In addition, this tool can be utilized in the management of design requirements issues. this tool has not been evaluated experimentally yet (Shahid *et al*., 2011).

DevComplete is a traceability tool designed by SmartBear software. It offers a complete traceability for project tasks, requirements tracking and flaws for team agility improvement. DevComplete permits freedom to implement requirement modifications according to current demands. It also detects the effects of additional software requirements. In addition, the tool is capable of enforcing the creation and modification of a specific traceability link. If any alteration occurs on a requirement, the traceability information may remain constant. Furthermore, the tool reads requirements and then imports them from requirement specification documents generated outside the tool while supporting the generation of descriptive documents for the requirements within the tool (Shahid *et al*., 2011).

With regards to a systematic literature review, this study includes a highlight on the practices that are used in RT, due to its great importance in the requirements management. The practice is able to improve product quality and monitor efficiently the changes of requirements during software development phases. The findings of the systematic review reveal the existence of a set of practices that are used to trace requirements, where the general trend of these practices are either automatic or semi-automatic rather than manual methods. They are beneficial in order to reduce the efforts in creating, maintaining and updating links and relationships traceability, reduce the time and efforts required to follow the requirements, overrun the mistakes that occur if tracing manually as well as difficulties of traceability in large projects that contain a large number of requirements.

There are varying levels of automation between the traceability techniques and tools. In regards to this, this study found three main levels of the automation; manual, semi-automatic and automatic. Manual level refers to practices that require human efforts to establish and maintain traceability relations. Meanwhile, semi-automatic level refers to automatic methods that still need human activities to monitor the results produced by automated methods and to provide feedback and retrace. In contrast, automated level refers to practices that generate traceability relations as a result of the software development process. For the manual level, RTM is an example of this level, used to define the relationship between requirements and other artifacts like design modules, code modules and test cases. It creates a link between requirements and other artifacts manually; therefore, they suffer from several problems, e.g., more prone to errors and also time-consuming.

Most of the techniques provided by the systematic review are semi-automatic level. The practices that are semi-automatic require manual effort (human analysts) to monitor the results produced by these methods. These practices include: value-based, scenario-based, hypertext-based, event-based, ArchTrace, goal-centric, model-based, RETRO and traceMAINTAINER. With regards to the automatic level, this group of automatic

traceability practices generates traceability relations as a result of the software development process and includes both IR and RB. Despite the speed, automatic traceability techniques in general, suffers from some disadvantages such as creating traces inaccurate in some cases or miss traces and generation of traceability links that are hard to understand and manage (Mirakhorli and Huang, 2011). For both DevComplete and DesignTrack, literature does not specify the level of automation and with any technique used.

CONCLUSION

This study presents the SLR of RT practices in the period from 2008-2015. The findings showed a set of RT practices that are used to create traceability links, maintenance of traceability relationships and update and evaluation of traceability links. Overall, this study was conducted with the aim of helping software companies to implement effective traceability in their software development and extracting a set of tools and techniques that assists them to accomplish their tasks.

SUGGESTIONS

More research in the area of requirements traceability within the academic setting is required in order to enhance the understanding of the influence of requirements traceability towards software development. Meanwhile, it is also recommended more studies that include applying these practices on different projects. The future research goal is to investigate the practices of requirement traceability applied among software companies in Malaysia. To do this, we will prepare a set of questionnaire that includes a series of questions about the use of these techniques in software companies.

ACKNOWLEDGEMENT

The researchers wish to thank the Ministry of Education Malaysia for funding this study under Fundamental Research Grant Scheme (FRGS), S/O project:-12818.

REFERENCES

Ahmad, A. and M.A. Ghazali, 2007. Documenting requirements traceability information for small projects. Proceedings of the IEEE International Conference on Multitopic, December 28-30, 2007, IEEE, Lahore, Pakistan, ISBN:978-1-4244-1552-6, pp: 1-5.

Athira, B. and P. Samuel, 2011. Traceability matrix for regression testing in distributed software development. Proceedings of the International Conference on Advances in Computing and Communications, July 22-24, 2011, Springer, Berlin, Germany, ISBN:978-3-642-22713-4, pp: 80-87.

Casey, V. and F. McCaffery, 2011. Med-Trace: Traceability assessment method for medical device software development. Proceedings of the Conference on European Systems and Software Process Improvement and Innovation EuroSPI, June 27-29, 2011, Roskilde University, Roskilde, Denmark, pp: 1-12.

Cleland-Huang, J., R. Settimi, O. BenKhadra, E. Berezhanskaya and S. Christina, 2005. Goal-centric traceability for managing non-functional requirements. Proceedings of the 27th International Conference on Software Engineering, May 15-21, 2005, ACM, St. Louis, Missouri, ISBN:1-58113-963-2, pp: 362-371.

Cuddeback, D., A. Dekhtyar and J.H. Hayes, 2010. Automated requirements traceability: The study of human analysts. Proceedings of the 18th IEEE International Conference on Requirements Engineering (RE), September 27-October 1, 2010, IEEE, Sydney, Australia, ISBN:978-1-4244-8022-7, pp: 231-240.

Dyba, T., B. Kitchenham and M. Jorgensen, 2005. Evidence-based software engineering for practitioners. Software IEEE., 22: 58-65.

Egyed, A. and P. Grunbacher, 2002. Automating requirements traceability: Beyond the record & replay paradigm. Proceedings of the 17th IEEE International Conference on in Automated Software Engineering ASE, September 23-27, 2002, IEEE, Edinburgh, England, ISBN:0-7695-1736-6, pp: 163-171.

Gotel, O.C.Z. and A.C.W. Finkelstein, 1994. An analysis of the requirements traceability problem. Proceedings of the 1st IEEE International Conference on Requirements Engineering, April 18-22, 1994, IEEE, Colorado Springs, Colorado, ISBN:0-8186-5480-5, pp: 94-101.

Han, K., J. Youn, and J. Cho, 2014. A functional requirement traceability management methodology for model-based testing framework of automotive embedded system. Proceedings of the 3rd International Conference on Advances in Vehicular Systems, Technologies and Applications, June 22-26, 2014, IARIA Publisher, Seville, Spain, ISBN:9781632667472, pp: 46-51.

Hannay, J. E., D.I. Sjoberg and T. Dyba, 2007. A systematic review of theory use in software engineering experiments. IEEE. Trans. Software Eng., 33: 87-107.

Hayes, J.H., A. Dekhtyar, S.K. Sundaram, E.A. Holbrook and S. Vadlamudi *et al*., 2007. Requirements Tracing on target (RETRO): Improving software maintenance through traceability recovery. Innov. Syst. Softw. Eng., 3: 193-202.

Heindl, M. and S. Biffl, 2005. A case study on value-based requirements tracing. Proceedings of the 10th Conference and 13th Joint ACM SIGSOFT International Symposium on European Software Engineering and Foundations of Software Engineering, September 05-09, 2005, ACM, Lisbon, Portugal, ISBN:1-59593-014-0, pp: 60-69.

Huang, J.C., C.K. Chang, G. Sethi, K. Javvaji and H. Hu *et al*., 2002. Automating speculative queries through event-based requirements traceability. Proceedings of the IEEE Joint International Conference on Requirements Engineering, September 9-13, 2002, IEEE, Essen, Germany, ISBN:0-7695-1465-0, pp: 289-296.

Huang, J.C., J.H. Hayes and J.M. Domel, 2009. Model-based traceability. Proceedings of the ICSE Workshop on Traceability in Emerging Forms of Software Engineering, May 18, 2009, IEEE, Vancouver, British Columbia, ISBN:978-1-4244-3741-2, pp: 6-10.

Huang, J.C., O. Gotel, J.H. Hayes, P. Mader and A. Zisman *et al*., 2014. Software traceability: Trends and future directions. Proceedings of the ACM Conference on Future of Software Engineering, May 31-June 07, 2014, ACM, Hyderabad, India, ISBN:978-1-4503-2865-4, pp: 55-69.

Kirova, V., N. Kirby, D. Kothari and G . Childress, 2008. Effective requirements traceability: Models, tools and practices. Bell Labs Tech. J., 12: 143-157.

Kitchenham, B. and S. Charters, 2007. Guidelines for performing systematic literature reviews in software engineering. MSc Thesis, Keele University, Keele, England.

Klimpke, L. and T. Hildenbrand, 2009. Towards end-to-end traceability: Insights and implications from five case studies. Proceedings of the 4th IEEE International Conference on Software Engineering Advances ICSEA'09, September 20-25, 2009, IEEE, Porto, Portugal, ISBN:978-1-4244-4779-4, pp: 465-470.

Lin, J., C.C. Lin, J.C. Huang, R. Settimi and J. Amaya *et al*., 2006. Poirot: A distributed tool supporting enterprise-wide automated traceability. Proceedings of the 14th IEEE International Conference on Requirements Engineering, September 11-15, 2006, IEEE, Minneapolis-Saint Paul, Minnesota, ISBN:0-7695-2555-5, pp: 363-364.

Mader, P. and A. Egyed, 2014. Do developers benefit from requirements traceability when evolving and maintaining a software system?. Empir. Softw. Eng., 20: 413-441.

Mader, P., O. Gotel and I. Philippow, 2009. Trace maintainer: A tool for the semi-automated maintenance of model traceability. Master Thesis, University of Twente, Enschede, Netherlands.

Mader, P., O. Gotel, T. Kuschke and I. Philippow, 2008. Trace maintainer-automated traceability maintenance. Proceedings of the 16th IEEE International Conference on Requirements Engineering RE'08, September 8-12, 2008, IEEE, Barcelona, Spain, ISBN:978-0-7695-3309-4, pp: 329-330.

Maletic, J.I., E.V. Munson, A. Marcus and T.N. Nguyen, 2003. Using a hypertext model for traceability link conformance analysis. Proceedings of the International Workshop on Traceability in Emerging Forms of Software Engineering, October 6-7, 2003, October 6-7, 2003, pp: 47-54.

Mirakhorli, M. and J.C. Huang, 2011. Tracing architectural concerns in high assurance systems (NIER track). Proceedings of the 33rd IEEE International Conference on Software Engineering, May 21-28, 2011, IEEE, Honolulu, Hawaii, ISBN:978-1-4503-0445-0, pp: 908-911.

Murta, L.G., A. van der Hoek and C.M. Werner, 2008. Continuous and automated evolution of architecture-to-implementation traceability links. Autom. Software Eng., 15: 75-107.

O'connor, A.M., K.M. Anderson, C.K. Goodell and J.M. Sargeant, 2014. Conducting systematic reviews of intervention questions I: Writing the review protocol, formulating the question and searching the literature. Zoonoses Publ. Health, 61: 28-38.

Ooi, S.M., R. Lim and C.C. Lim, 2014. An integrated system for end-to-end traceability and requirements test coverage. Proceedings of the 5th IEEE International Conference on Software Engineering and Service Science (ICSESS), June 27-29, 2014, IEEE, Beijing, China, ISBN:978-1-4799-3278-8, pp: 45-48.

Ozkaya, I. and O. Akin, 2007. Tool support for computer-aided requirement traceability in architectural design: The case of design track. Autom. Constr., 16: 674-684.

Petticrew, M. and H. Roberts, 2008. Systematic Reviews in the Social Sciences: A Practical Guide. John Wiley & Sons, Hoboken, New Jersey,.

Ramesh, B. and M. Jarke, 2001. Toward reference models for requirements traceability. Softw. Eng. IEEE. Trans., 27: 58-93.

Regan, G., F. McCaffery, K. McDaid and D. Flood, 2012. The barriers to traceability and their potential solutions: Towards a reference framework. Proceedings of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), September 5-8, 2012, IEEE, Cesme, Izmir, ISBN:978-1-4673-2451-9, pp: 319-322.

Regan, G., F. McCaffery, K. McDaid and D. Flood, 2014. The development and validation of a traceability assessment model. Proceedings of the International Conference on Software Process Improvement and Capability Determination, November 4-6, 2014, Springer, Vilnius, Lithuania, ISBN:978-3-319-13035-4, pp: 72-83.

Rempel, P., P. Mader and T. Kuschke, 2013. An empirical study on project-specific traceability strategies. Proceedings of the 21st IEEE International Conference on Requirements Engineering (RE), July 15-19, 2013, IEEE, Rio de Janeiro, Brazil, ISBN:978-1-4673-5765-4, pp: 195-204.

Reshef, N.A., B.T. Nolan, J. Rubin and Y.S. Gafni, 2006. Model traceability. IBM. Syst. J., 45: 515-526.

Rosmadi, N.A., S. Ahmad and N. Abdullah, 2015. The Relevance of Software Requirement Defect Management to Improve Requirements and Product Quality: A Systematic Literature Review. In: Pattern Analysis, Intelligent Security and the Internet of Things, Abraham, A., A.K. Muda and Y.H. Choo (Eds.). Springer, Cham, Switzerland, ISBN:978-3-319-17397-9, pp: 95-106.

Salvador, C., A. Nakasone and J.A.P. Sang, 2014. A systematic review of usability techniques in agile methodologies. Proceedings of the 7th Euro American Conference on Telematics and Information Systems, April 02-04, 2014, ACM, Valparaiso, Chile, ISBN:978-1-4503-2435-9, pp: 1-17.

Schwarz, H., J. Ebert and A. Winter, 2010. Graph-based traceability: A comprehensive approach. Softw. Syst. Model., 9: 473-492.

Shahid, M., S. Ibrahim and M.N. Mahrin, 2011. An evaluation of requirements management and traceability tools. World Acad. Sci. Eng. Technol., 78: 596-601.

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

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