Novateur Publication's
International Journal of Innovation in Engineering, Research and Technology [IJIERT]
ICITDCEME'15 Conference Proceedings
ISSN No - 2394-3696

A SURVEY ON ACCURACY OF REQUIREMENT TRACEABILITY LINKS DURING SOFTWARE DEVELOPMENT

Mr. Vinayak M. Sale,
Department of Computer Science & Technology,
Shivaji University, Kolhapur, IndiaEmail- csvs13510@gmail.com

Prof. Santaji K. Shinde
Department of Information Technology,
Bharati Vidyapeeth's College of Engineering, Kolhapur, India
Email: santaji@rediffmail.com

ABSTRACT
Traceability is used to ensure that source code of a system is consistent with its requirements. The only specified requirement has been implemented by developers. During software maintenance and evolution, requirement traceability links become marginal because no developer can devote effort to update it. However, to recover traceability links later is a very painful and monotonous task also it is costly for developers too. Traceability supports the software development process in various ways, like as change management, software maintenance and prevention of misunderstandings. But, while, in practice, traceability links between requirements and codes are not created during the development of software as it requires extra efforts. So developers rarely use such links during development. Why many challenges exist in traceability practices today? However, many of the challenges can be overcome through organizational policy, quality requirements traceability tool support remains the open problem.

KEY WORDS: Traceability, requirement, management

INTRODUCTION

The traceability is very most important for any software project, and if we use it, it could be beneficial from different perspectives for the development. When we develop a source code for any system that source code can be traced and become identical with the requirement and analysis because we develop a source code as per the requirement. A traceability link is the connection between the source code and requirement. Requirement traceability helps software engineers to trace the requirement from its emergence to its fulfillment [5]. Traceability may not help us to know how different components of systems are interlinked and dependent on each other in the same system. We may also fail to find the impact of change on the software and system. A most important goal of traceability, in absence of original requirements and other artifacts traceability links. Therefore, we should look at traceability from all the aspects of traceability regarding scope and coverage [1].

Requirements traceability has proved much important over the past decade in the scientific literature. It is defined as "the ability to illustrate and go after the life of a requirement, in both a onward and backward direction". Traceability links among of a system and its source code helps us in reducing system comprehension attempt. While updating the software, the developers can add, remove, or modify features as per the users demand. While maintenance and evolution of any software, requirement traceability links become marginal because no developer can devote effort to update it. However to recover traceability links later is a very painful and tedious task also it is costly for developers too. In fact, developers usually do not update requirement-traceability links with source code. Requirements and source codes are different from each other, which decreases the textual similarity [2].

REASONS FOR REQUIREMENTS TRACEABILITY

The traceability is one of needs of stakeholders – project sponsors, project managers, analysts, designers, maintainers, and end users, because of their need, priority, and goal. The requirements traceability is a characteristic of a system in which the requirements are clearly linked to their sources and the artifacts formed during the system development life cycle based on these requirements [15].

In requirements engineering and elicitation phase, it is important that the rationales and sources to the requirements are captured to know requirements development and confirmation [15].

Modifications in design appear e.g. if the requirements evolve or if the system is developed incrementally [14]. During design phase requirements traceability helps to keep track of when the change request is implemented before a system is redesigned. Traceability is able to impart information about the justifications, important decisions and assumptions related to requirements [15].

Modifications after the delivery of the system occur due to various reasons (e.g. to a changing environment). Such modifications are called system evolution [11] Empirical studies show that even experienced software professionals predict incomplete sets of change impacts [17]. With the help of complete traceability, more accurate cost and schedule of change(s) can be determined, instead of depending on the engineer or programmer who is expert [15].

A SURVEY OF RELEVANT LITERATURE

Traceability recovery, feature location, and trust models topics are related to this research work. Traceability approaches can be divided into three main categories, i.e., dynamic, static, and hybrid.

Dynamic traceability approaches [9] require a system to be compliable and executable to perform traceability creation tasks. It also requires pre-defined scenarios to execute the software system.

Dynamic approach collects and analyzes execution traces [9] to identify method a software link has been executed in the particular scenario. However, it couldn't help to differ in overlapping scenarios, because a single method has some limitations. Due to bugs and/or some other issues the legacy system may not be applicable. Thus, to collect execution traces is not possible.

Static traceability approaches [8], [14] use source code structure and/or textual information to recover traceability links between high-level and low-level software artifacts.

Software repositories have been preferred by many researchers [9] to recover traceability links, presented a formula based approach to recover traceability links between software artifacts in which software systems' version history is taken into consideration. It assumes, two files must have a potential link between them if they co-change. However, in the certain case, two files are co-changing but they do not have any semantic relationship. Also, it is possible that some software artifacts do not have software repositories, in such a case, their approach cannot find the link from/to these documents, e.g., requirement specifications. In hybrid traceability, [4] there is a combination of static and dynamic information. The study shows that combination of dynamic and static information can perform better than the single IR technique.

The results are achieved by static approaches show that they do not require an executable software system. Thus, static traceability approaches can be applied to a system that contains a bug or is not executable.

DIFFICULTIES IN REQUIREMENT TRACEABILITY

Apart from the benefits that traceability offers to the software engineering industry, there are many difficulties in practice. These difficulties can be identified at the cost of time and effort, the difficulty of maintaining traceability through change, different views on traceability held by various project stakeholders, organizational problems and politics, and poor tool support.

COST: One of the biggest challenges facing the implementation of traceability is the cost involved. As a system grows in size and complexity, capturing the requirement traces quickly becomes complex and expensive [10]. Because of this, the budget of a project gets collapsed. However, traceability can be detected in early development process when it is easy and cheap too.

By this one can save a significant amount of effort by focusing traceability activities on the most important requirements. However, it requires a clear understanding of each requirement in the system. It may not be an option if full tracing is a requirement of the customer or the development process standards used for the project.

On the other hand, the high costs of traceability that is incurred can save much bigger sum in future of software projects. It cannot solve the problem of the high costs of traceability implementation, but considering long term benefits this incurred cost is beneficial because it can save a lot.

MANAGING VARIATIONS: To maintain traceability in different situations is another challenge. Experts assume that change is inevitable in the life of any matter and software project is not objection too. If there is in the change, then you have to update the traceability data to reflect such change [12]. To update the traceability data, the separate system is required, which can be costly as it requires much of time if the change is extensive.

However, that discipline cannot be universal and applicable for all changes under every circumstance. To deal with change and its impact on traceability is not a cup of tea. Some tools can be useful to identify the impact of change on existing traceability data; but, still it requires a lot of efforts to update it [13]. At the same time, training can help users to understand the importance of discipline in maintaining traceability data.

By keeping the eye at long-term benefits, developer prefers short-term incurred costs to sustain the organization.

TYPES OF TRACEABILITY

Over the years, several other terms related to requirements traceability have been established. According to Winkler & von Pilgrim [6], the most common ones are pre-requirements specification, post-requirements specification, forward, backward, horizontal, and vertical traceability. These terms are shown in Figure 1 and described in detail in the following.

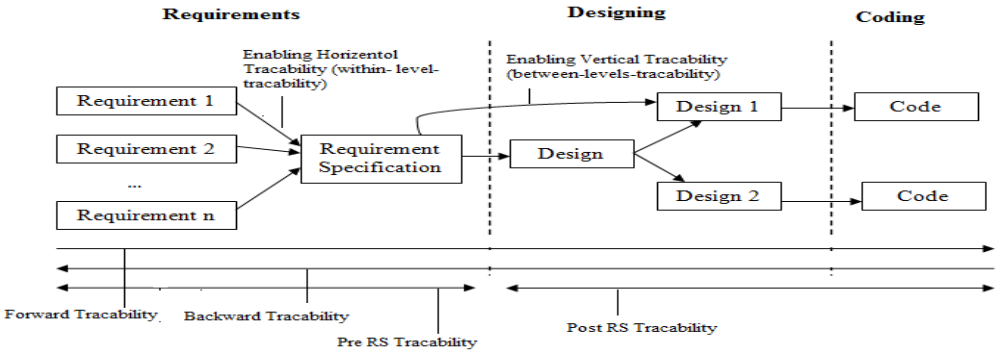

Fig. 1 Different Types of Traceability

Gotel & Finkelstein [19] have introduced the classification of pre-requirements specification (pre-RS) traceability and post-requirements (post-RS) traceability. Pre-RS traceability is related to those aspects of a requirement's life before its inclusion in the RS, which means all traces that occur during elicitation, discussion, and agreement of requirements. This includes dealing with informal, conflict, or overlap of information [6]. Post-RS traceability is concerned to such aspects of a requirement's life that resulted from its addition in the RS, which means all traces that occur during the stepwise implementation of the requirements in the design and coding phases. It includes documenting the traces of the various manual and automatic transformation steps eventually producing the system [6].

The SRS has introduced the terms backward traceability and forward traceability. Backward traceability refers to the ability to follow the traceability link from particular artifact to its sources from which is has been derived. Forward traceability stands for following the traceability links to the artifacts that have been derived from the artifact under construction.

Ramesh & Edwards [20] have introduced the terms horizontal traceability and vertical traceability. These terms are used for the traceability links of an artifact belonging to the same project phase or level of abstraction (horizontal), and links between artifacts belonging to different ones (vertical) [6].

Instead of above definitions Winkler & von Pilgrim [6] defines these essential traceability links as follows:

1. Traceability means the ability to describe and follow the life of a software artifact in the sense of the generalized definition presented by [19].

2. A trace is a piece of (implicit or explicit) information which is an indication or evidence showing what has existed or happened.
3. Finally, a traceability link is, as already stated, a relation that is used to interrelate artifacts (e.g., by causality, content, etc.) Following the notation of a trace, a traceability link is a more concrete (but not the only) form of information that can be used to describe and follow certain aspects of the life of the representative software artifacts [16].

REPRESENTING TRACEABILITY

Firstly, to any software evolution task, a developer has to comprehend the project landscape [4], particularly, system architecture, design, implementation and the relations between the various artifacts using any available document. Program comprehension occurs in a bottom-up manner, a top-down manner, or some combination thereof [3]. Developers use different types of knowledge during program comprehension, ranging from domain-specific knowledge to general programming knowledge. Traceability links between source code and sections of the documentation, e.g., requirements, aid both top-down and bottom-up comprehension [1].

Traceability links between the requirements of a system and its source code are helpful in reducing comprehension effort. Requirement traceability is defined by [4] [5], "the capability to demonstrate and go subsequent to the life of a requirement, in both a forward and toward the back direction". This traceability information also helps in software maintenance and evolution tasks.

In order to use traceability links, it is necessary to represent them in a form that is appropriate for its purpose. Several different ways (traceability matrices, graphical models, cross references) exist to represent traceability links, which are also supported by tools. Wieringa [18] distinguishes between three different kinds of traceability representation, while [7] represent artifacts and the traceability links between them as a graph:

- **TRACEABILITY MATRICES:** Traceability links are represented in matrix form. The horizontal and vertical dimensions are linked. The entries in the matrix represent links between the artifacts in the matrix [18].
- **GRAPHICAL MODELS:** Entity Relationship Model (ERM) is also used to represent traceability links. Also, various UML diagrams support the representation of traceability links embedded in the different development models [18].
- **CROSS REFERENCES:** Traceability links between artifacts are represented as links, pointers or annotations in the text [18].

CONCLUSION

To develop any software requirements, traceability plays vital role similarly it plays the vital role in the maintenance of software. Creating traceability links manually is one of the costly laborious work. Still it is need of the time to make efforts on traceability links more cheaper in short standard solution should be formed. Requirements specification for requirements traceability is formed alongside all the investigations, which drives both their direction and focus.

REFERENCES

[1] Prashant N. Khetade, Vinod V.Nayyar "Establishing a Traceability Links Between The Source Code And Requirement Analysis, A Survey on Traceability", Int'l Conf on Advances in Engg & Tech – 2014 (ICAET-2014) 66 | Page (IOSR-JCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727 PP 66-70

[2] S. Muthamizharasi, J. Selvakumar, M.Rajaram "Advanced Matching Technique for Trustrace To Improve The Accuracy Of Requirement" Int'l Journal of Innovative Research in Science, Engg, and Tech -(ICETS'14) Volume 3, Special Issue 1, February 2014

[3] N. Ali, Y.-G. Gue´he´neuc, and G. Antoniol, "Trustrace: Mining Software Repositories to Improve the Accuracy of Requirement Traceability Links" IEEE Trans. Software Eng., vol. 39, no. 5, pp. 725-741, May 2013

[4] N. Ali, Y.-G. Gue´he´neuc, and G. Antoniol, "Trust-Based Requirements Traceability", Proc. 19th IEEE Int'l Conf. Program Comprehension, S.E. Sim and F. Ricca, eds., pp. 111-120, June 2011.

[5] N. Ali, Y.-G. Gue´he´neuc, and G. Antoniol, "Factors Impacting the Inputs of Traceability Recovery Approaches", A. Zisman, J. Cleland-Huang, and O. Gotel, eds. Springer-Verlag, 2011.

[6] Winkler, S., & Pilgrim, J. A survey of traceability in requirements engineering and model-driven development. Software & Systems Modeling, vol. 9, issue 4, pp. 529-565 (2010)

[7] Schwarz, H., Ebert, J., and Winter, A. Graph-based traceability: a comprehensive approach. Software and Systems Modeling (2009)

[8] J. H. Hayes, G. Antoniol, and Y.-G. Gue´he´neuc, "PREREQIR: Recovering Pre-Requirements via Cluster Analysis," Proc. 15th Working Conf. Reverse Eng., pp. 165-174, Oct. 2008.

[9] D. Poshyvanyk, Y.-G. Gue´he´neuc, A. Marcus, G. Antoniol, and V. Rajlich, "Feature Location Using Probabilistic Ranking of Methods Based on Execution Scenarios and Information Retrieval," IEEE Trans. Software Eng., vol. 33, no. 6, pp. 420-432, June 2007.

[10] Heindl, Matthias, and Stefan Biffl. A Case Study on Value-Based Requirements Tracing. Proc. of the 10th European Software Engineering Conference. Lisbon, Portugal, 2005: 60-69

[11] Lehman, M., Ramil, J. Software Evolution – Background, Theory, Practice Information Processing Letters, Vol. 88, Issues 1-2, October 2003, pages 33-44

[12] von K nethen, A .Change-Oriented Requirements Traceability. Support for Evolution of Embedded Systems Proc. of International Conference on Software Maintenance, October 2002, pages 482-485

[13] Cleland-Huang, Jane, Carl K. Chang, and Yujia Ge. Supporting Event Based Traceability Through High-Level Recognition of Change Events. Proc. of the 26th Annual International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment. Oxford, England, 2002: 595-602.

[14] G. Antoniol, G. Canfora, G. Casazza, A.D. Lucia, and E. Merlo, "Recovering Traceability Links between Code and Documentation," IEEE Trans. Software Eng., vol. 28, no. 10, pp. 970-983, Oct. 2002.

[15] Ramesh, B., Jarke, M. Toward Reference Models for Requirements Traceability IEEE Transactions on Software Engineering, Vol. 27, No. 1, January 2001, pages 58-93

[16] Clarke, Siobhán, et al. Subject Oriented Design: Towards Improved Alignment of Requirements, Design, and Code. Proc. of the 1999 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications. Dallas, TX: 325-329.

[17] Lindvall, M., Sandahl, K. How well do experienced software developers predict software change? The Journal of Systems and Software 43, 1998, pages 19-27

[18] Wieringa, R. An Introduction to Requirements Traceability. Technical Report IR-389, Faculty of Mathematics and Computer Science (1995)

[19] Gotel, O. & Finkelstein, A. An analysis of the requirements traceability problem. In Proceedings of the First Int'l Conf. on Requirements Engineering, pp. 94-101 (1994)

[20] Ramesh, B., Edwards, M. Issues in the development of a requirements traceability model. In Proceedings of the IEEE International Symposium on Requirements Engineering, pp. 256-259 (1993)