An Emperical Study of Performance in the Requirement Traceability Links (RTL) Using Initial Mapping Technique

T. Hemalatha[1] N. Prakash[2], S. Sathyaraj S[3]

[1] P.G. Student, [2] Assistant Professor, [3] Assistant Professor

Department of Computer Science and Engineering

Oxford College of Engineering and Technology

Abstract - Requirement Traceability Link (RTL) in software engineering refers the ability of describing and following the life of a requirement in both forward and backward direction. While software development it is very essential to ensure that the traceability links for each and every requirement in the project. From the software requirement specification document the traceability goes through design document and coding in the forward direction. During software maintenance and evolution, requirement traceability links become obsolete because developers do not devote effort to updating them. Yet, recovering these traceability links later is a daunting and costly task for developers. To avoid such issues in software project management the developer should keep updating the traceability in design document and software requirement specification. Here two completed projects are taken in the educational domain to study the efficiency and accuracy of the requirement traceability first all the requirements are categorized using the decision tree algorithm. And then all the requirements are arranged in an order using merge sort according to its severity. To maintain the links in the backward direction that means while doing any change in the coding it is really a tough task to maintain the link through design document and software requirement specification (SRS) document. This backward traceability is achieved using a machine learning and trained initial mapping algorithm to improve the efficiency and accuracy. Then comapared the result of the existing system with the new technique used in this work and found that the proposed model is producing better results.

***Index Terms*— Requirement Traceability, Requirement Elicitation, Machine Learning, Software Requirement Specification, Sorting, Categorising**

1. INTRODUCTION:

The requirement is considered to be the most important part of software engineering process of a system or a product. Requirements are those which uniquely discover the various required attributes, features, functionalities and the quality of a software system. Among the various list of attributes of requirements the most predominant and important one is that it must be "traceable". Traceable in general confirms with the point that the requirement is authoritatively documented and satisfies all or part of the required needs as stated by the clients or the stakeholders. Requirement Engineering is defined as a systematic way to gather and analyze the requirements which are consecutively subjected to various processes like problem analysis, requirement collection, analysis and validation. Requirement engineering consists of two important phases as:

- Requirement Elicitation.
- Requirement Analysis.

A successful software engineering process depends upon a high quality requirement management.Requirement elicitation refers to the task of communicating with clients through various methods and gathers all the requirements efficiently. It uses various techniques such as questionnaire, brainstorming, group discussion, interview and prototyping. On the other hand, the requirement analysis corresponds to the task of determining whether the mentioned requirements are clear, complete and resolves if any issues arises.

2. REQUIREMENT TRACEABILITY:

Requirement traceability refers to documenting the life of a requirement and providing traceability between these requirements and other different attributes concerned in the phases of the development. Requirement traceability is considered to be an important discipline of requirement management in software development and system engineering.

In general it is denoted that it is the ability to describe and follow the life of a requirement in both forward and backward directions. Tracing is basically the concept of identifying all parts of the software system from requirements and the ability to trace back from the product to the requirements. Requirement traceability allows tracing the requirements from the instance it is obtained until it is implemented into a running coed as a specification. The different objectives of a requirement traceability are:

- Capable of managing the changes efficiently.
- Understand the software product under development and its functionality.
- Bridges the gap between the software and the environment in which it operates and provides consistency among them.

3. METHODS AVAILABLE IN SOFTWARE REQUIREMENT TRACEABILITY:

3.1. Traceability Links:

Traceability links are one of important and predominant method of requirement traceability. It corresponds to the tracking of the relationship between each requirement and its

origin. Traceability links are meant for tracking the relationship between each requirement and the end product to which the requirement is allocated.There are four typical types of traceability links as described in figure Fig 1. They are:

Forward to requirement links:

This keeps the tracking information from the sources of the requirements up to their analysis.

Backward from requirement links:

This link helps in tracking the information from requirement to their corresponding origin.

Forward from requirement links:

This link records the information of the progress from the requirement for the product.

Backward to requirement links:

This link maintains the data used in the tracking of the final product to their corresponding initial requirement. Here the forward to requirement and backward from requirement links are together denoted as a pre-requirement traceability links whereas, the rest two are termed as post-requirement traceability links with respect to the direction of the flow in relation to the requirement. These traceability links are required to bi-directional. Bi-directional traceability links give the capability to analyze the impact of the modification where all the products are affected by a modification in the requirement and all the requirements are affected by a modification or a fault in the products. Traceability links in general provide continuous assessment of the current status of the requirements and the end products by determining the missing requirements.

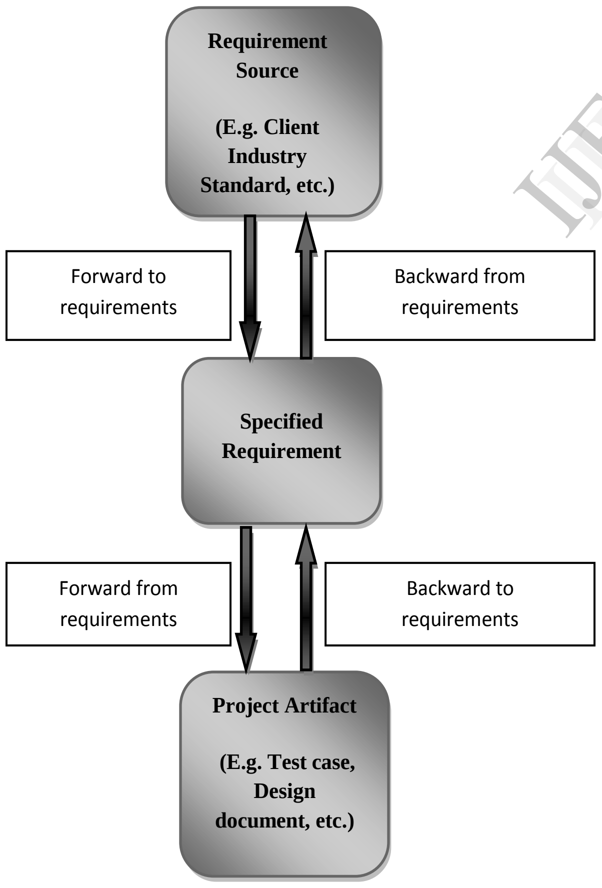
Requirement Source
(E.g. Client Industry Standard, etc.)
Forward to requirements
Backward from requirements
Specified Requirement
Forward from requirements
Backward to requirements
Project Artifact
(E.g. Test case, Design document, etc.)

Fig 1. Requirement traceability link's description.

3.2. TRACEABILITY MATRICES:

A traceability matrix is considered to be a document in the form of a table that correlates two baseline documents and measure the completeness of the relationship. It is also considered to be a classical tool to ensure that the project’s objective, scope, requirements and end products remain same when compared to the baseline document.This is a rarely used method of requirement traceability. Non-functional requirements such as performance goals, objectives and quality measures don’t always trace into the code. In general, the functional requirements are traced backwards to their parent non-functional requirement and forward to the products.

Among these various methods of traceability this paper concentrates on the method of requirement traceability links. In a concise, these requirement traceability links helps a lot in holding every information right from the source along with the requirements till the development process ends and also throughout the maintenance phase. If this is the case it would be the very useful way through which it is possible to maintain a complete record of the product generation. But the key point lies with regular updating of these links which is often left out by the developers and later becomes a very tedious process.

4. TECHNIQUES USED IN UPDATING THE REQUIMENT TRACEABILITY LINKS

With the importance of the requirement traceability links in mind the literature has proposed various techniques that automatically or semi-automatically update these links. Among the lots of available techniques two which we considered are:

4.1. TECHINQUE 1:

This technique proposes a trust based approach where the main concept was in mining software repositories and combining those mined results with Information retrieval (IR) techniques which would improve the accuracy of RTLs with subject to better recall and precision values. In general, the IR techniques can automatically recover RTLs.

4.2. TECHINQUE 2:

This technique has been framed with the objective to improve experts trust on a recovered link and trust over traceability inputs. It constitutes of three basic components such as LTI, TFC and HTA.

LTI: This uses various sources of information to increase the expert.

TFC: This helps in finding out which factors impact traceability process inputs and document them in a trust pattern.

HTA: This process combines different traceability recovery approaches.

5. Empirical Evaluation

Like above the literature has proposed various approaches for requirement traceability. But those were not completely successful and satisfactory for both the directions. They were very efficient in improving the accuracy of the links in the forward direction alone. But, a successful software developer must incorporate traceability to be bi-directional. This paper provides a solution for this problem wherein we suggest a model for detecting the modification of requirements also in the reverse engineering process and enhance the efficiency of the RTLs.

6. Experimental Work

For the experimental work we have taken two completed project in the education domain and find the performance of the Though the literature has enlisted with lots of techniques and approaches for recovering the traceability links and also updating the recovered links, still there are lots of drawbacks and inefficiencies in most of the softwares developed. One of the main reasons in correspondence with RTL is that most of the techniques are well suited and efficient in the forward direction. But, when the situation arises in a place where reverse engineering takes place or when it is a need to track behind from the coding of a module to its corresponding design document or requirement it fails to be successful. The problem is not only the inefficiency of the techniques used but also with the developers as they forget in updating the links regularly. This drawback will automatically result in reduced efficiency of the software being produced. Also, there will be problems such as wrong estimation, increased cost, complexity, delay in the development and increased rate of bugs and errors which on a whole decreases the quality of the software product. In order to reduce such complexities that harm the quality of the software product an approach that helps in traceability along the reverse direction also. This in turn will comparatively increase the efficiency of the product at a very high rate. Such a technique is being discussed in this paper.

If in such a way that when the requirement become possible to be accessed in backward direction also then this will automatically reflect the changes back up to the initial document of Software requirement specification (SRS).

6.1. INITIAL MAPPING WORK:

As described in the diagram Fig 2. The following steps are carried out in the initial mapping work.

Step 1: Initially all the requirements of the clients are properly gathered and analyzed using different elicitation and analysis techniques.

Step 2: Each of the gathered requirements is named uniquely like R1, R2, R3...… RN (where n denotes the last number of the requirements).

Step 3: Similarly proper investigation is made and the corresponding detailed design of the requirements R1 up to Rn are names as DD1, DD2, DD3 ….. DDn.

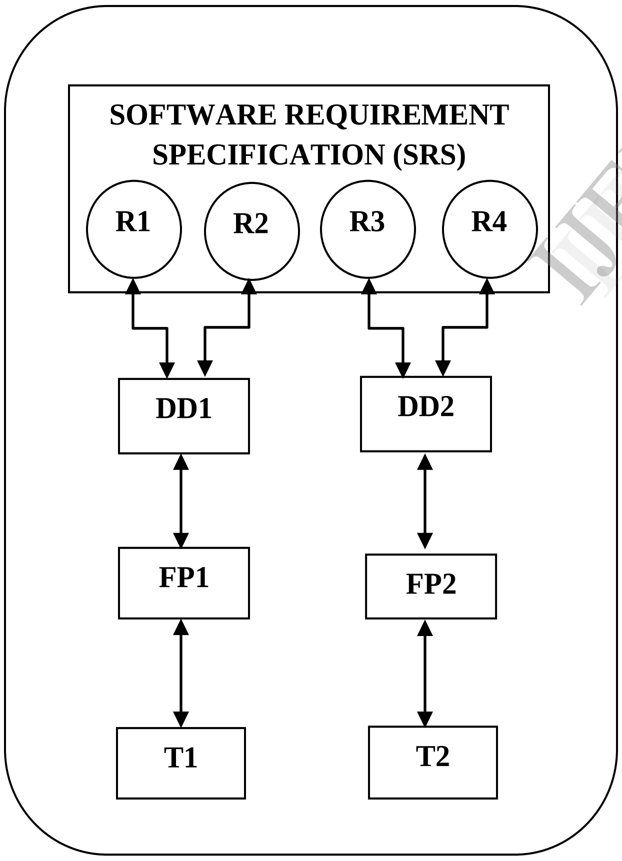

Fig 2. Initial mapping technique

Step 4: Here the requirements in a group like can depend on a single design document. For example, R1, R2 and R3 can be dependent on DD1; R4, R5 can be dependent on DD2 and so on.

Step 5: For all the above named DD1 to DDn their corresponding functional points are to be identified and noted as FP1, FP2, FP3 …. FPn.

(DD1, DD2 – Design documents,

FP1, FP2- Functional points of source code,

T1, T2 – Test cases.)

Step 6: Here also a single or a group of DDs can depend on a particular functional point. For example, DD1 depends on FP1, DD2, DD3 and DD4 depends on FP2 and so on.

Step 7: Then the corresponding test cases for their respective modules are taken as T1, T2, T3, …. Tn. Here test cases can be framed separately for single or group of modules.

Step 8: After identification of the entire component items proper mapping of every requirement in their progress must be noted which helps in tracking whenever necessary.

Such a way the comaparision of the two techniques

6.2. PROVISION OF LINK:

The output of the requirement analysis phases is generally a Software requirement specification (SRS). This SRS in turn are fed to the design phase where it generates the design document. These when entering the coding phase is developed in the source code and as a test case in the testing phase so on and so forth. Here the links for every requirement in each of these outputs according to which it is being used are to be given. This would rather help in maintaining the traceability paths.

6.3. MODIFICATION DETECTION SYSTEM:

Once links are given it is to be designed in such a way that a change when made highlights it in the mapped requirements. This helps in enhancement of modification detection system. Let us consider that in the development of a project let R1, R2 and R3 be three initial requirements and DD1 be their corresponding design document and FP1 be their corresponding functional point. Then, if a change or error occurs in FP1 the system would automatically highlight it corresponding DD1 and R1, R2 and R3 respectively.

When this happens then it is possible to manage the entire traceability easily. Also if any change occurs during the coding phase then it correspondingly highlights the requirements and design document which could be later used for reference. Also when an error or a bug arises in any functional module then with reference to the highlighted portions it could be easier to identify the correct part where the error could be recovered.

In order to check the working and accuracy of this system we developed a project for result analysis in the college. Then we took it as a sample for evaluation of the system and the result data is tabulated in Table 1 and are used for generation of the following graph (Fig 3).

From the graph (Fig 3) a conclusion can be derived that the proposed system has detected comparatively a large number of changes or modifications than the related works in improving the accuracy of requirement traceability links. Another notable point is that the related works highly detects and corrects the modifications only in the coding phase whereas the proposed system detects equally in all the phases.

SOFTWARE ENGINEERING PHASES	ACCURACY OF DETECTION IN THIS WORK (%)	ACCURACY OF DETECTION IN RELATED WORKS (%)
REQUIREMENT ANALYSIS	69.2	49.1
DESIGN	60.1	52.1
CODING	78.9	88.1
TESTING	73.1	65.1

Table 1. Tabulation of data obtained from a sample project subject to the proposed system in comparison with other related work.

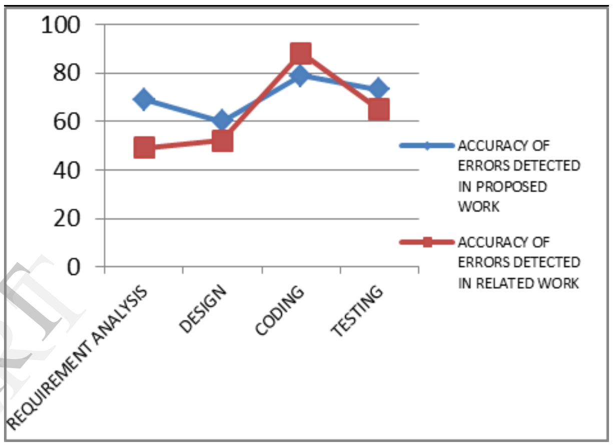

Fig 3. Graph denoting the difference in accuracy of errors between the proposed and related system with the above data in the table.

7. CONCLUSION AND FUTURE WORK:

This paper provides a experimental for the detection of modifications and errors in the requirement traceability links. Here a mapping is being provided between the outputs of the various phases carried out during the development process of the software. This is designed in such a way that the error occurred in any part of the development would automatically trace its path in the remaining phases and would denote that part by highlighting the phrases.

This is designed in such a way that it is possible to make the traceability links accessible in both forward and backward directions. In general, the errors detected in the testing phase are recovered only in the source code alone in a fast manner. But this system would rather help to identify the problem right from the requirement source that would eventually eradicate the problem completely from arising in the future.

The future work of this paper is in concentration with updating the detected modifications made and creating and maintaining a separate repository to store these modifications so that it can be used in the future.

8. REFERENCES:

[1] Nasir Ali, Member, Yann-Ga¨ El Gu´eh´eneuc, and Giuliano Antoniol, “Trustrace: Mining software repositories to improve the accuracy of requirement traceability links”, *IEEE Transactions on Software Engineering*, 29 Oct. 2012. IEEE computer Society Digital Library.

[2] O. C. Z. Gotel and C. W. Finkelstein, “An analysis of the requirements traceability problem,” Requirements Engineering., Proceedings of the First International Conference on, pp. 94–101, April 1994.

[3] N. Ali, Y. -G. Gu´eh´eneuc, and G. Antoniol, “Trust-based requirements traceability,” in Proceedings of the 19th International Conference on Program Comprehension, S. E. Sim and F. Ricca, Eds. IEEE Computer Society Press, June 2011, 10 pages.

[4] G. Antoniol, G. Canfora, G. Casazza, A. D. Lucia, and E. Merlo, “Recovering traceability links between code and documentation,” IEEE Transactions on Software Engineering, vol. 28, no. 10, pp. 970– 983, 2002.

[5] A. Marcus and J. I. Maletic, “Recovering documentation-to source-code traceability links using latent semantic indexing,” in Proceedings of 25th International Conference on Software Engineering. Portland Oregon USA: IEEE CS Press, 2003, pp. 125–135.

[6] Murphy, G. C., Notkin, D. And Sullivan, K., Software Reflexion Models: Bridging the Gap between Source and High-Level Models, In the Proceedings of the Third ACM SIGSOFT Symposium on the Foundations of Software Engineering, October 1995, ACM, New York, NY, p. 18-28.

[7] Ramesh, Bala; Stubbs, Lt Curtis; & Edwards, Michael. "Lessons Learned from Implementing Requirements Traceability." Crosstalk, Journal of Defense Software Engineering 8, 4 (April 1995): 11-15.

[8] Ramesh, B., Jarke, M., Toward Reference Models for Requirements Traceability, IEEE Transactions On Software Engineering, vol. 27, no. 1, January 2001.

[9] INCOSE requirements management tool survey, Online at http://www.incose.org.

http://www.stsc.hill.af.mil/crosstalk/1995/apr/Lessons.asp