International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 3, March - 2014

Improve the Accuracy of Requirement Traceability Links using Requirement Severity

M. Johara
M.E.(Computer Science and Engineering)
University College of Engineering-Panruti

T. Hemalatha
M.E.(Computer Science and Engineering)
Oxford Engineering College

***Abstract:-* The most important intention being determined in this work is an concern about the Requirement Traceability Links (RTL) that is the lack of updating of those links. Requirement Engineering refers to the process of formulating, documenting and maintaining software requirements and to the subfield of Software Engineering concerned with this process. Requirement traceability is concerned with documenting the life of a requirement and providing bi-directional trabceability between various associated requirement. It enables user to find the origin of each requirement and track every change that was made to this requirement. It has been argued that even the use of the requirement after the implemented features have been deployed and used should be traceable. From the lots of available traceability methods this paper considers the usage of RTL during the development of a software system. During the stages of software development and maintenance as insufficient effort is being taken by the programmers in updating the RTLs, they become inexperienced and at the later stage it becomes very difficult and costly process for doing so. Hence, this paper proposes an automated system that detects all the modifications that takes place at all the stages of development for every requirement gathered initially. The main objective of this paper is detecting the modification of requirements in the RTL during the reverse engineering process also.**

1. INTRODUCTION:

A requirement is a singular documented physical and functional need that a particular design, product or process must be able to perform. It is most commonly used in a formal sense in system software engineering (or) enterprise engineering. It is a statement that identifies a necessary attribute, capacity, characteristics, or quality of a system for it to have value and utility to a customer, organization, internal user, or other stakeholder. Requirements also an important input in to the verification process since tests should trace back to specific requirements. Among the various list of attributes of requirements the most major and important one is that it must be "traceable". Traceable in general confirms with the point that the requirement is confidently documented and satisfies all or part of the required needs as stated by the clients or the stakeholders.

Requirement Engineering emphasizes the use of systematic and repeatable technique that ensure the completeness, consistency, and relevance of the system requirements. Requirement engineering consists of two important phases as:

- Requirement Elicitation.
- Requirement Analysis.

The term elicitation is used to raise the fact that good requirements cannot just be collected from the customer, as would be indicated by the name requirements gathering. Requirement elicitation is the practice of collecting the requirements of a system from users, customers and other stake holders. Requirement elicitation is non-trivial because you can never be sure you get all requirements from the user and customer by just asking them what the system do. Requirement elicitation practices include interviews, questionnaires, user observation, workshops, brainstorming, use cases, role playing and prototyping. Before requirements can be analyzed, modeled, (or) specified they must be gathered through an elicitation process. Requirement elicitation is the part of the requirements engineering process, usually followed.

2. REQUIREMENT TRACEABILITY:

Traceability is the ability to verify the history, location, or application of an item by means of documented recorded identifiers. It include to capability of keeping track of a given set (or) type of information to a given degree, (or) the ability to chronologically interrelate uniquely identifiable entities in a way that is verifiable.Requirement traceability is concerned with documenting the life of a requirement and providing bi-directional traceability between various associated requirements. It enables users to find the origin of each requirement and track every change that was made to this requirement. For this purpose, it may be necessary to document every change made to the requirements. It is concerned with documenting the relation ships between requirement and other development artifacts. Its purpose is to facilitate:

- The overall quality of the product[s] under development;

- The understanding of product under development and its artifact; and
- The ability to manage change.

3. METHODS OF REQUIREMENT TRACEABILITY:

3.1. Traceabiltiy 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.

The ability to perform requirements tracing can be accomplished by four different types of links, utilizing both forward and backward direction tracing. The different links that canbe created for requirement tracing are:

1. Forward From The Requirements
2. Backward To The Requirements
3. Forward To The Requirements
4. Backward From The Requirements

3.1.1. Forward From The Requirements:

The link “Forward From The Requirements” can be described by assigning a requirement to one or more system components. Once this is done, then the component (or) components are “responsible” for the requirement. This type of link is good for evaluating impact of requirement change

3.1.2. Backward To The Requirements:

The link “Backward To The Requirements” is done when the compliance of the system or a system component is mapped back to the requirements. This type of link will show that the requirement has been tested (or) verified for the system.

3.1.3. Forward To The Requirements:

The link “Forward To The Requirements” can be described by taking the customers needs and technical assumptions and linking them to a requirement or requirements. If the customers needs or a technical assumption changes, then the requiremets can be analyzed for impact.

3.1.4. Backward From The Requirements:

The link “Backward From The Requirements” will give the ability to validate the requirement or requirements by the customers needs or technical assumptions.

Customer needs are traced forward to requirements, So you can tell which requirements, will be affected if those needs change during or after development. This also gives you confidence that the requirements specification has addressed all stated customer needs to identify the origin of each software requirement. If you presented customer needs in the form of use cases, the top half of Figure 1 illustrates tracing between use cases and functional requirements. The bottom of Figure 1 indicates that, as requirements flow into down stream deliverables during development, you can trace forward from requirements by defining links between individual requirements and specific product elements. This types of link assures that you have satisfied every requirement because you know which components address each one.The fourth types of link traces specific work products backward to requirements so that you know why each item was created. Most applications include code that doesn’t relate directly user-specified requirements, but you should know why someone wrote every line of code.

Suppose a tester discovers unexpected functionality with no corresponding written requirement. This code could indicate that a developer implemented a legitimate implied requirement that the analyst can now add to the specification. Alternatively, it might be “orphan” code, an instance of gold plating that doesn’t belong in the product. Traceability links can help you sort out these kinds of situations and build a more complete pictures of how the pieces of your system fit together. Conversely, test cases that are derived from-and traced back to individual requirements provide a mechanism for detecting unimplemented requirements because the tester wont find the expected functionality

Traceability links help you keep track of parentage, interconnections, and dependencies among individual requirements. This information reveals the propagation of change that can result when a specific requirement is deleted or modified. If you have mapped specific requirements into tasks in your projects work-breakdown structure, those tasks will be affected when a requirement is changed or deleted. There are four typical types of traceability links as described in figure Fig 1. They are:

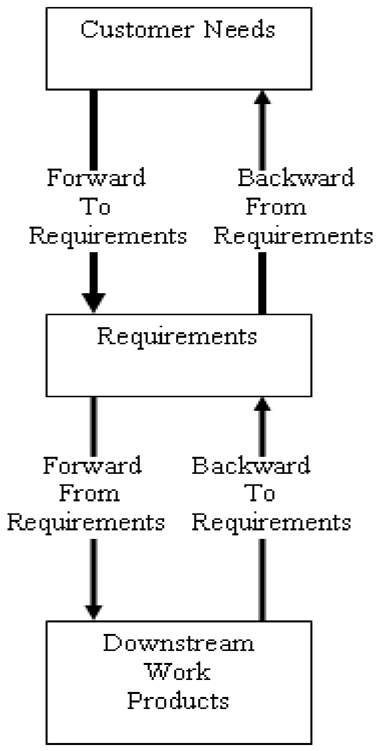

Fig 1. Requirement traceability link's description.

3.2. Traceability Matrices:

Traceability matrix links a business requirement to its corresponding functional requirement right up to the corresponding test cases. If a Test Case fails, traceability helps to determine corresponding functionality easily. It also helps ensure that all requirements are tested. It is a method used to validate the compliance of a process or product with the requirements for that process or product. The requirements are each listed in arrow of the matrix and the columns of the matrix are used to identify how and where each requirement has been addressed.In a software development process, a traceability matrix is a table that correlates any two baselined documents that require a many to many relationship to determine the completeness of the relationship. It is often used with high level requirements and detailed requirement of the software product to the matching parts of high level design, detailed design, test plan and test cases

3.3. Traceability And Non-Functional Requirements:

Traceability is a bidirectional process between stated requirements both functional and non-functional requirements and the completeness of the product in meeting those requirements.

Unlike the functional requirements that support a certain degree of locality in the system, non-functional requirement, being related to the system quality apply to the overall qualities of the system. In most cases, non-functional requirements provide solutions in pattern and are applied to the system in the design phase. If the traceability between the analysis model elements, design model elements and the elements involved in the architectural pattern is not maintained in the application process, it may be very costly to reflect the changes of the non-functional requirements in the system .This study process the mechanism in which the traceability between anlysis model, design model and pattern elements is set in real time while applying the architecture pattern simultaneously to minimize the costly change of non-functional requirements.

4. EXISTING TECHNIQUES IN UPDATING RTLs:

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:

IR based approaches are very well suited to address the traceability recovery problem.

In general, the IR techniques can automatically recover RTLs.

4.2. Technique 2:

This technique automatically manage traceability link evolution and update the links in evolving software. It constitutes of three basic components such as TLR, LSI, and TLEM.

TLR: Provide the support for automatic traceability link evolution management that can cope with software evolution.

LSI: Automatic link updating technique relies on a novel incremental version of the well-known Laten Sematic Indexing that have been used for TLR

TLEM: During software development automatically maintain and update the links.

5. PROBLEM STATEMENT:

The source code of a system is not consistent with its requirement and that all and only the specified requirement have been implemented by developers. During software maintanence and evolution, requirement traceability becomes outdated because developers do not/cannot devote effort to updating them. Yet, recovering thse traceability links later is a discouraging and costly task for developers. But a successful software developer must recover these traceability links semiautomatically or automatically

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. PROPOSED WORK:

The proposed work support automated traceability maintanence by recognizing development activities. Development activities are formally specified and changes to certain model elements trigger a LinkUp dateManager. This manager is responsible for updating traceability links that are related to the changed element. IR technique that uses latent semantic indexing for feature location. It shows combining different approaches can perform better than IR technique. It helps developers to keep source code identifiers and comments consistent with high level artificat. It uses traceability from requirements to surce code and uses bug reports, mailing lists, temporal information and so on, as reputation trust for a traceability link. As the reputation of a link increases, the trust in this link also increases..

Here the following methods are to be carried out for success of the objective.

6.1. Initial Mapping Work:

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 tracability goes through design document and coding in the forward directon.

First all the requirements 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 specification. This backward traceability achieved using a machine learning and initial mapping algorithm to improve the efficiency and accuracy. The compared the result of the existing with the new technique used in this work and found that the proposed model producing better results.

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

Step 3: Then the corresponding test cases for their respective modules are taken.Here test cases can be framed separately for single or group of modules.

Step 4: Similarly proper investigation is made and the corresponding detailed design of the requirements.Here the requirements in a group like can depend on a single design document. .

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.

6.2. Traceability System:

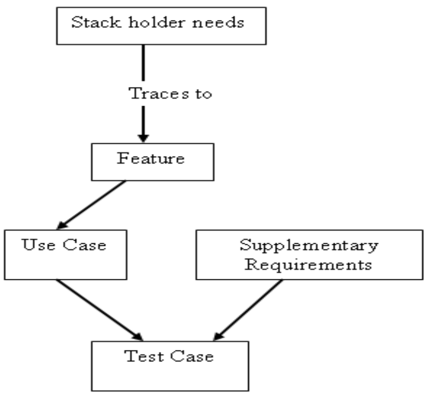

Fig 2: Total Traceability System

Fig 2. Illustrates User needs will be traced to features. Features will be traced to use case. Any needs not traced to a features will not be implemented. Use case will be traced to test cases. Supplemental specifications will be traced to test case.

6.3. Modification Detection System:

Requirements tracinc can improve the quality of products, reduce maintanence cost and facilitate reuse. Following are some potential benefits of implementing requirements traceability.

1. Certification
2. Change impact analysis
3. Maintanence
4. Project tracking
5. Re-engineering
6. Reuse
7. Risk reduction
8. Testings

Many of these long-term benefits, reducing overall product life-cycle costs but increasing the development cost by effort you expend to accumulate and manage the traceability information. 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.

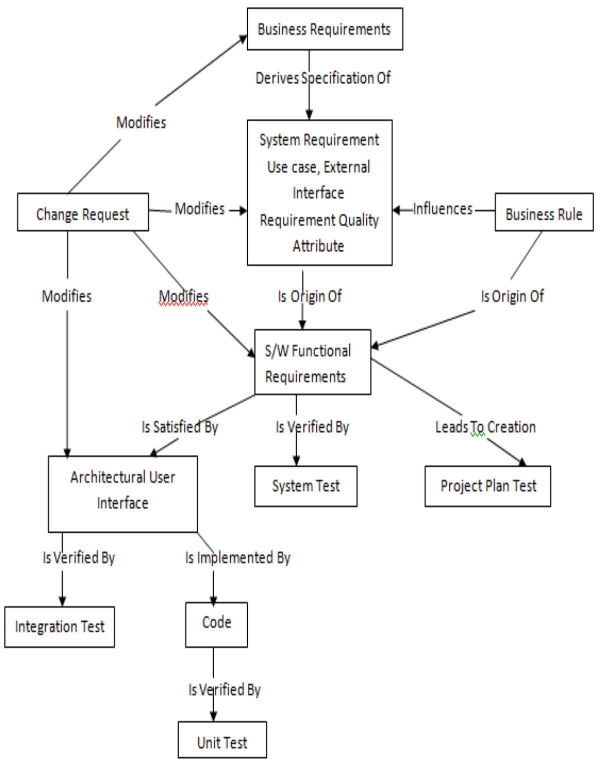

Fig 2: Modification Detection System

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.

PHASES	ACCURACY OF CHANGES DETECTED IN PROPOSED WORK	ACCURACY OF CHANGES DETECTED IN RELATED WORK
REQUIREMENT ANALYSIS	68.3	48.2
DESIGN	59.2	51.2
CODING	77.8	87.1
TESTING	72.2	64.2

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

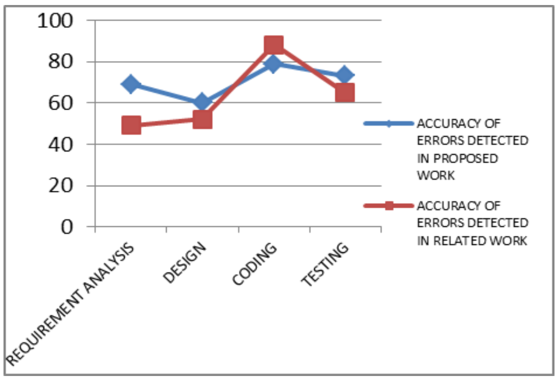

Fig 3. Graph denoting the difference in accuracy of errors between the proposed and related system with the above data in the table.

7. CONCLUSION AND FUTURE WORK:

Thus, this paper provides a contribution 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. S.A. Sherba and K.M. Andrson, “A Framework for Managing Traceability Relationships between Requirements and Architectures,” Proc. Second Int’l Software Requirements to Architectures Workshop, part of Int’l Conf.Software Eng., pp.150-156, 2003
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. M.Gethers, R.Oliveto, D.Poshyvanyk, and A.D.Lucia, “On Integrating Orthogonal Information Retrieval Methods to Improve Traceability Recovery,” Proc. 27th IEEE Int’l Conf. Software Maintanence, pp.133-142, Sept.2011.
8. Ramesh, B., Jarke, M., Toward Reference Models for Requirements Traceability, IEEE Transactions On Software Engineering, vol. 27, no. 1, January 2001.
9. A.Marcus and J.I. Maletic, “Recovering Documentation-to-Source-Code Traceability Links Using Latent Semantic Indexing, Proc.Int’l Conf.Software Eng., pp.125-135, May 2003
10. P.Mader, O. Gotel, and I. Philippow, “Enabling Automated Traceability Maintanence by Recognizing Development Activities Applied to Models,” Proc.23rd IEEE/ACM Int’l conf. Automated Software Eng., pp. 49-58,2008