ICEST-2K24 [International Conference on Engineering, Science and Technology]
In Association with
International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN - 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)

A Review Paper for Accuracy of Requirement Traceability Links in Software Development

Mr. Vinayak M. Sale[1], Ms. Samruddhi Pramod Kaldhone[2], Ms. Pranjali Jagannath Dekhane[2]

[1]Assistant Professor Department of Artificial Intelligence & Data Science, FTC COER, Sangola, Maharashtra, India

[2]B.Tech Students Department of Artificial Intelligence & Data Science, FTC COER, Sangola, Maharashtra, India

ABSTRACT

Requirement satisfaction is a vital aspect in the execution of software. The requirements are identified by the different stakeholders should be fulfilled with each point of the development of the software. Software development is a correlated organizational work to automate continuous liberate of new software development while assurance their accuracy and consistency. For tracing the requirement from its starting point to its completion,

requirement traceability supports software engineers to trace. During software improvement process, traceability supports in a different of ways, like change management, software maintenance, and prevention of confusion. But, many of the challenges can be reduce through organizational policy, quality requirements traceability tool support remains the open problem. During software updating and maintenance; the traceability links become out-of-date since the developers can modify or remove some features of the source code.

Keywords: requirements, traceability, management, requirement traceability approach (RTA), IR Technique (IRT)

I. INTRODUCTION

Essentially, for any developing any software a developer must identify the project background, in particular, the system architecture, design, working, and the relations between the several components using any available documentation[1] [6]. Program idea occurs in a bottom-up method, a top-down method or some combination of both.

The traceability is the most essential factor for development of any software project, and if we use it, it could be valuable from different perspectives for the software development. While developing any software, we develop source code which can be traced and become identical with the requirement and analysis because we develop a source code as per the requirements. [2] [3] A traceability is an association between the source code and requirement.

Requirement traceability supports software engineers to trace the requirement from its development to its

fulfillment [2] [7]. Traceability may not help us to know how different components of systems are inserted and dependent on each other in the same system. We may also fail to find the impact of change on the software and system [4]. An important objective of traceability is a linkage of, in the lack of original requirements and other artifacts traceability links [3]. Therefore, we should look at traceability from all the aspects of traceability regarding scope and coverage.

While modernizing the software, the developers can add, remove, or modify features as per the users' request. While software maintenance and development, requirement traceability links become fringe because any developer can't devote effort to update it. Conversely, for recovering traceability links later is a very painful and tedious task also it is costly for developers too [6]. A developer usually does not update requirement-traceability links with source code.

Requirements and source codes are different from each other, which decrease the textual similarity [2] [4].

II. REASONS FOR REQUIREMENTS TRACEABILITY

It is most important to confirm that the requirements are properly fulfilled in the design. This is done with requirements traceability which is usually referred to as [5] [6] [7] [18] "the ability to validate and go after the life of a requirement, in both forward and the back direction." [23] Requirements traceability confines the relationships between the requirements and source code. The traceability is one of the needs of different stakeholders – project sponsors, project managers, analysts, designers, maintainers, and end-users, because of their need, priority, and goal [1][3] [6].

During design phase requirements traceability supports to keep track of when the changes are implemented before a system is redesigned. Traceability can also give information about the validation, significant-conclusion, and postulation behind requirements [2] [18].

After the delivery of the system, [1] modifications occur due to various reasons (e.g. to a changing environment). The traceability helps us complete, more accurate cost and schedule of change(s) can be resolute, instead of depending on the engineer or programmer who is expert [18].

Traceability information allows answering:

1. What is the outcome, when the requirements are changed?
2. Where is a requirement useful?
3. Are all requirements assigned?
4. Which require is deal with by a requirement? 5. Is this requirement essential?
5. What design decisions affect the implementation of a requirement?
6. What are the benefits of this technique and what were the further options?
7. Is the implementation compliant with the requirements?
8. Is this design element necessary?
9. How do I interpret this requirement?

Benefits of traceability

1. Stops losing of knowledge
2. Supports for the verification process
3. Change control
4. Process monitoring
5. Better software quality

6. Reengineering
7. Reusability
8. Decrease of Risk

III.BASIC TRACEABILITY LINKS

Traceability links depend on the traceability information, the linking of maybe

1. *One-to-one* -one design element to one code module
2. *One-to-many* - one functional requirement verified by multiple test cases
3. *Many-to-many* - a use case may lead to multiple functional requirements, and a functional requirement may be common to several use cases

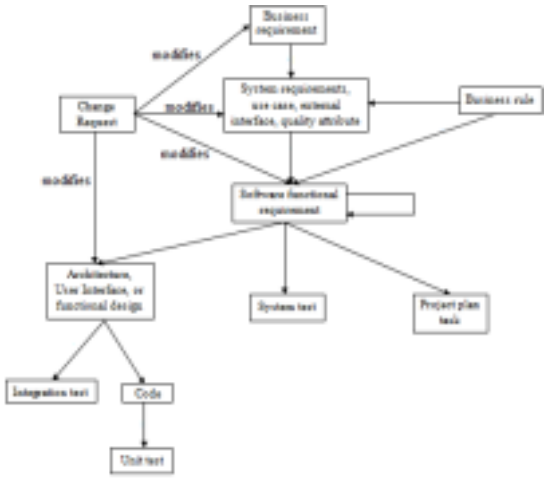

Fig 1 Example of traceability links

IV.BACKGROUND AND RELATED WORK

This section presents an environment on the IR technique and a review of the related work. Traceability approach can be separated into three main categories, i.e., dynamic, static, and hybrid.

The dynamic approach gathers and examines execution traces [11] to recognize the technique that a software link has been carrying out in the particular scenario. However, it couldn't help to differ in overlapping circumstances, because there are some limitations to a single method. [6] The legacy system may not be applicable, due to bugs and/or some other issues. Thus, to collect execution traces is not possible.

Static traceability approaches [10], [17] use source code structure and/or textual information for recovering traceability associations among high-level and low-level software artifacts. The combination of static and dynamic information is hybrid traceability. The study shows that a combination of dynamic and static information can perform better than the single IR technique [7].

INFORMATION RETRIEVAL TECHNIQUE (IRT)

Information Retrieval (IR) refers to a method that would calculate textual similarities of different documents. The textual similarity is calculated using the terms that occurred in the documents. If two documents have a number of general terms, those documents are measured to be similar. The analysis of different IR methods can be in three steps [14]. First, after pre-processing such as stop word removal and stemming, a corpus is made from the documents.

Second, each document is represented as access in an index. The term-by-document matrix is a common index, where the document as rows and each term as a column. The incidence of the term arising in the document is the values in the matrix. Third, by using a cosine similarity formula, the similarity among the index entries is calculated [24]. The presentation of the key entries and the formula for calculating the similarity varies depends on the IR method. We use the VSM IR method in this paper and briefly describe it in the following paragraph.

In the Vector Space Model (VSM) [14] the vector of terms is represented by each document. In the term-by-document matrix, each row can be measured as one document's vector in the space of terms that occur in all documents. The calculation of similarity of two documents is based on the cosine angle between vectors of each document. In general, the cosine angle between vectors of the two documents will reduce as the different documents share more terms. Hence, the higher similarity of the documents will occur.

V. SYSTEM ARCHITECTURE

1. DESIGN OF SYSTEM

For any software evolution, the essential task is, a developer must understand the project background [5] [6], in particular, the system planning, propose, how to implement, and the relations among the different artifacts using any available documentation. Program understanding occurs in a bottom-up way, a top-down way, or some mixture thereof. Different types of data, ranging from domain-specific

knowledge to general programming knowledge can be used throughout program conception [2]. Traceability links between source code and part of the documentation, e.g., requirements, abet both top-down and bottom-up conception.

Requirement traceability is defined as, "the capability to demonstrate and go after the life of a requirement, in both onward and toward the back direction" [23]. Traceability links are also necessary to make sure that source code is reliable with its requirements and that all and only the specified requirements have been implemented by developers.

Traceability links are useful in decreasing understanding effort between the requirements of a system and its source code [3] [4] [7]. The traceability information is also useful for software maintenance and development tasks. For instance, once a developer has traceability links, a user can easily trace what software artifacts must be changed for the development of a new requirement.

Even with the importance of traceability links, in software maintenance and development, as developer update features, requirement traceability links become outdated because developers do not dedicate effort to update them later [4] [5] [6]. This lacking traceability information is one of the main issues, that contribute to project failure, and difficult to sustain. Unsatisfactory traceability information results in the need for costly and painstaking tasks of manual recovery and maintenance of traceability links. [23] These manual tasks may be frequently required depending on how normally software systems evolve or are maintained.

As a result, the literature proposed methods, techniques, and tools to improve automatically traceability links. [5] [6] [22] [23] Researchers used information retrieval (IR); techniques, to recover traceability links between high-level documents, e.g., requirements, instruction booklet pages, and plan documents, and low-level documents, e.g., source code and UML diagrams. IR techniques compute the textual similarity between each two software artifacts, e.g., the source code of a class and a requirement. [2] [3] [5] [6] [7] A high textual similarity means that the two artifacts most likely share numerous concepts and that; therefore, they are likely

linked to one another.

2. SYSTEM BLOCK DIAGRAM

The proposed work is based on the IR-based RTAs process is typically divided into three main steps. Figure 1 shows the IR-based RT links revival process.

First, every the textual information with the requirements and source code is taken out and preprocessed by splitting terms, [2] [3] [7] removing stop words and remaining words are then stemmed to its grammatical origin. Second, all the stemmed terms are weighted using a term weighting system. Last, an IR technique calculates the similarity between

requirements and source code documents. [4] Lastly, it creates a ranked list of probable traceability links. An elevated comparison between two documents shows a probable semantic connection between them.

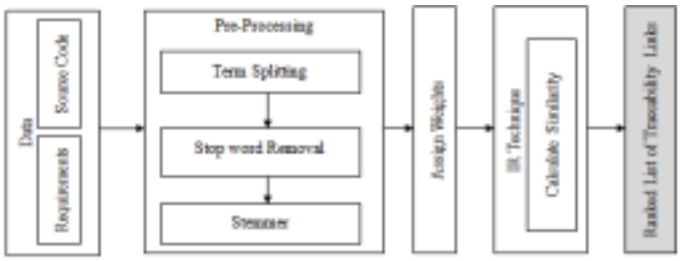

Fig 1. System Block Diagram

2.1 PRE-PROCESSING

To generate traceability links, we remove all the identifiers from source code and terms from requirements. In this, IR techniques are used as an engine to create links between requirements and source code. IR techniques imagine that all documents are in the textual format [5] [9]. To remove source code identifiers, a source code parser is used. The parser throw-outs extra information, e.g., primary data types and keywords, from the source code and gives only identifier names. The removal of the identifiers and terms is followed by filtering, stopper, and stemmer process [11] [12].

The primary step is term splitting. [2] [3] [6] A text normalization step renovates all upper-case letters into lower-case letters. This step eliminates non-textual, i.e., some numbers, mathematical symbols, brackets, etc., information and extra white spaces, from the documents. Some identifiers/terms could be united with some special characters, e.g., underscore, and/or Camel Case naming reunion. Therefore, divide all the united terms to make them separate. For example, Hello India and hello india are split into the terms “hello india” [6].

The following step is the stop word removal. [5] [6] The input for this step is the normalized text that could include some general words, e.g., articles, punctuation, etc. These general words are measured as noise in the text because it does not be a symbol of the semantics of a document. Hence, in this step, a stop word list is used to eliminate all the stop words.

The next step is stemming. An English stemmer, for example, would recognize the terms “excel,” “excellence,” and/or “excellent” as based on the root “excel” [6] [7]. An IR technique calculates the similarity between two documents based on similar terms in both documents. Still, due to different postfix, IR techniques would judge them, e.g., add and addition are like two different documents, and the result would be a low similarity between two documents. Thus, it becomes important to perform the morphological investigation to exchange plural into the singular and to take back infected forms to their morphemes [2].

Following two main factors are considered important [6]:

Term frequency (TF): TF is often called home frequency. If a term appears several times in a document, then it

would be allocated higher TF than the others.

Global frequency (GF): If a term appears in various documents then the term is considered global. It is also known as inverse document frequency (IDF).

2.2 TERM WEIGHTING / ASSIGN WEIGHTS

An IR technique (IRT) changes all the documents into vectors to calculate the similarities along with them. [16] [17] To change documents terms into vectors, each term is allocated a weight. A variety of schemes for weighting terms have been proposed in the literature. Widely used weighting schemes are differentiated as probabilistic. In the following, the term identifiers to refer all source code entities, i.e., class name, method name, variable name, and comments [7].

If a term comes out multiple times in a single or multiple documents, then IRT would propose that document as a relevant document to a query [5] [7] [10]. However, multiple amounts of a term do not show that it is an important term.

2.3 IR TECHNIQUES

To create sets of traceability links, various IR techniques are used, to identify concepts in the source code, carry out experiments using different IR techniques to recover traceability links [4] [5].

PROPOSED ALGORITHMS

1. DATA PREPROCESSING

The data preprocessing is prepared to eliminate needless content from the text and to find out the origin form of the words. [2] [7] The preprocessing of the data is completed by a valid method such as Stop word removal and Stemming to the data composed of the customer.

2. STOP WORD REMOVAL:

For work out, stop words are words that are filtered out preceding to, or following, processing of text. [2] [3] Stop words are ordinary words that take less significant meaning than keyword. These stop words are a few of the most common, short function words, such as the, a, an, is, at, which, that, and on, etc.

Stop-word elimination is the method of eliminating these words. To find out the words from a text all needless content must be removed, so it is needed to remove the stop words from the text put into an array [7] [17].

Algorithm:

1. The following is an algorithm for stop word removal 1.Acquire the input
2. Establish the glossary of stop words
3. Divide factors into words
4. Assign new word list to store words
5. Collect outcome in the String Builder
6. Loop during the entire terms
7. Come again string with words detached

8. STEMMING

Words get from the input of the data are create to be too sparse to be useful as features for categorization as they do not simplify well. The presence of a large number of inflections of the same word, this is the common reason for stemming. Hence, the origin form of the word is to be taking out as a feature [2] [6] [7].

Stemming is the method, for decreasing derived words to their origin form. Stemming program is commonly known as stemming algorithms or stemmers [2].

Even as writing the sentence for a grammatical basis, it contains various forms of a word, for example, collect,

collection, collecting and/or collected. In many circumstances, it would be helpful for a finding for one of these words to revisit the word in the set to take away the required content from a given sentence [2] [7] [8].
The goal of stemming is to decrease variation form and sometimes derivationally related forms of a word to a common base form [5] [6] [7].
For instance: car, cars, -> car

Stemming algorithm:

The stemming algorithm consists of different steps of stemming applied sequentially. Within each stage, there are various principles to select rules, such as choosing the rule from every rule group that applies to the longest suffix. The algorithm of stemming works as follows:
Rules Illustrations
S → cats → cat
EED ->EE agreed →agree
(*v*) ED → plastered → plaster
(*v*) ING → cutting → cut
There are three main reasons for stemming algorithm, or stemmer, as follows.
The first reason of a stemmer is to cluster the words according to their theme. Many words are the root from the same stem, and we can consider that they belong to the same concept (e.g., act, actor, action). [2] [3] [5] The different forms are created by attaching affixes (prefixes, infixes, and/or suffixes) but, in English considering only suffixes, as normally prefixes and infixes change the meaning of the word, and a bit of them would lead to errors of bad topic resolve The next reason of a stemmer is openly associated to the [2] [3] [7] [10] IR process, as containing the stems of the words agree to some point of the IR process to be better, among which we can stress the ability to index the documents according to their theme, as their terms are clustered by stems or the extension of a query to obtain to a greater extent accurate results.
The extension of the query permits it, for refining by replacing the terms, it covers the related topics, which are also there in the collection [3] [5] [15]. This alteration can be done routinely and obviously to users, or the system can propose one or more superior method of the query
Finally, the conflation of the words allocation the same stem leads to a decrease of the vocabulary to be taken into the process, as the entire terms contained in the natural input collection of documents can be decreased to a set of topics [2] [4] [7]. This directs to a decrease of the space needed to store the formation used by an IR system and after that also lightens the computational weight of the system.

VI. HOW TO REPRESENT TRACEABILITY

Program conception occurs in a bottom-top way, a top bottom way, or a mixture of them [4] [8]. Developers use knowledge throughout program comprehension, from domain-oriented knowledge to common programming knowledge. Traceability linkage between source code and sections of the documentation, e.g., requirements, aid both top-down and bottom-up comprehension [1]. Traceability linkage between the requirements of a system and its source code is useful in reducing comprehension effort.
Requirement traceability is defined by [6] [23], “the capability to demonstrate and go after to the life of a requirement, in both onward and toward the back direction”. This traceability information also supports in software maintenance and evolution tasks. For traceability links, it is essential to represent them in a form that is suitable for its purpose [1]. The different ways (traceability matrices, graphical models, cross-references)

exist to represent traceability links, which are also supported by tools.

a. Traceability matrices: Traceability links are represented in matrix form. The traceability matrix is the association between, horizontal and vertical dimensions are the values in the matrix stand for links between the artifacts in the matrix [21].
b. Graphical models: Entity-Relationship Model (ERM), various UML diagrams support the representation of traceability links embedded in the different development models [21].
c. Cross references: Traceability associations between different parts are represented as links, pointers, or annotations in the text [21].

VII. CONCLUSION

The traceability is most important factor and precious from different point of views for the development for any software project. For development of any software, requirement traceability plays a vital role in the maintenance of software. Creating traceability links manually is one of the costly and lengthy works. Requirements specification for requirements traceability is formed alongside all the investigations, which drives both their direction and focus.

VIII. REFERENCES

[1]. Prof. Santaji K. Shinde, Mr. Vinayak M. Sale, "A Survey on Accuracy of Requirement Traceability Links During Software Development" Int'l Conf. on Innovations & Technological Developments in Computer, Electronics and Mechanical Engineering (ICITDCEME), 2015

[2]. Divya K.S., Dr. R. Subha, Dr. S. Palaniswami "Similar Words Identification Using Naive and TF-IDF Method" I.J. Information Technology and Computer Science, 2014, 11, 42-47 Published Online October 2014 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijitcs.2014.11.06 Copyright © 2014 MECS I.J. Information Technology and Computer Science, 2014, 11, 42-47

[3]. 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

[4]. 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

[5]. 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

[6]. 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.

[7]. 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.

[8]. 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)

[9]. Schwarz, H., Ebert, J., and Winter, A. Graph-based traceability: a comprehensive approach. Software and Systems Modeling (2009) [10] 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.

[10]. 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.

[11]. 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

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

[13]. 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, 2003, pp. 125– 135.

[14]. 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

[15]. 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.

[16]. 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.

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

[18]. 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.

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

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

[21]. 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)

[22]. 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)

[23]. Annibale Panichella, Collin McMillan, Evan Moritz, Davide Palmieri, "When and How Using Structural Information to Improve IR-based Traceability Recovery"

[24]. LEONARDO LEITE, CARLA ROCHA, FABIO KON, DEJAN MILOJICIC, PAULO MEIRELLES, "A Survey of DevOps Concepts and Challenges" 18 Nov 2019