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

Similar Words Identification Using Naive and TF-IDF Method

Divya K.S.
PG Scholar, Department of CSE, Sri Krishna College of Technology, Coimbatore, India
Email: divyaks.1990@gmail.com

Dr. **R. Subha**
Assistant Professor, Department of CSE, Sri Krishna College of Technology, Coimbatore, India
Email: kris.subha@gmail.com

Dr. **S. Palaniswami**
Principal, Government College of Engineering, Bodinayakanur, India
Email:joegct81@yahoo.com

***Abstract*—**Requirement satisfaction is one of the most important factors to success of software. All the requirements that are specified by the customer should be satisfied in every phase of the development of the software. Satisfaction assessment is the determination of whether each component of the requirement has been addressed in the design document. The objective of this paper is to implement two methods to identify the satisfied requirements in the design document. To identify the satisfied requirements, similar words in both of the documents are determined. The methods such as Naive satisfaction assessment and TF-IDF satisfaction assessment are performed to determine the similar words that are present in the requirements document and design documents. The two methods are evaluated on the basis of the precision and recall value. To perform the stemming, the Porter's stemming algorithm is used. The satisfaction assessment methods would determine the similarity in the requirement and design documents. The final result would give a accurate picture of the requirement satisfaction so that the defects can be determined at the early stage of software development. Since the defects determines at the early stage, the cost would be low to correct the defects.

***Index Terms*—**Requirements Document, Design Documents, Requirement Satisfaction, Porter's Stemming Algorithm, Term Frequency, Inverse Document Frequency

I. Introduction

Software Engineering is the study and application of engineering to the design, development, and maintenance of software [9]. Software engineering may also be defined as the systematic design and development of software products and the management of the software process. Requirements engineering (RE) refers to the process of formulating, documenting and maintaining software requirements and to the subfield of Software Engineering concerned with this process [10]. The major activities of requirement engineering are requirement elicitation, requirement analysis, requirement specification and requirement validation.

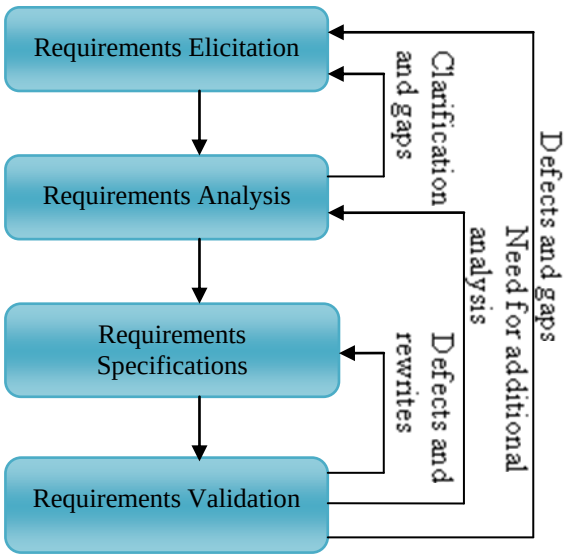

Fig. 1. Requirements Engineering activities

The requirements elicitation step includes all of the activities involved in identifying the requirement's stakeholders, selecting representatives from each stakeholder class, and determining the needs of each class of stakeholders. The goal of the elicitation activity is to improve the understanding of the requirements that is to achieve progress in the content dimension. During the elicitation activity, requirements are elicited from stakeholders and other requirement sources. Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirements analysis is an important aspect of project management. A Software requirements specification (SRS), a requirements specification for a software system, is a complete description of the behavior of a system to be developed and may include a set of use cases that

describe interactions the users will have with the software. In addition it also contains non-functional requirements. Requirements validation is the assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers.

The major problems found in requirement engineering [11] are the following:

- Poor Requirements Quality
- Over Emphasis on Simplistic Use Case Modeling
- Inappropriate Constraints
- Requirements Not Traced
- Missing Requirements
- Excessive Requirements Volatility including Unmanaged Scope Creep
- Inadequate Verification of Requirements Quality
- Inadequate Requirements Validation
- Inadequate Requirements Management
- Inadequate Requirements Process
- Inadequate Tool Support
- Unprepared Requirements Engineers

Satisfaction assessment is the determination of whether each component of the requirement document has been addressed in the design document. The steps of Satisfaction assessment[1] are the following:

- Analyzing a high level document and identifying the high level element
- Analyzing a low level document and identifying the low level element
- Determining whether each aspect of high level element have been addressed by the low level elements
- Assign a label to the high level element

The paper presents a satisfaction approach to identify the similarities between the requirements document and the design document. The documents are preprocessed to find outs the words that have least importance in the phrases. Porter's stemming algorithm is used for reducing derived words to their stems. The Naive satisfaction assessment and TF-IDF satisfaction assessment are performed to find the similarities in the documents.

According to this method, the rest of the paper is structured as follows. In Section 2, some related works are presented. In section 3, more details about the methodology of the proposed system are described. Finally in section 5 conclusions and some future works are discussed.

II. Related Works

Elizabeth Ashlee Holbrook, Jane Huffman Hayes, Alex Dekhtyar and Wenbin Li perform a study of methods for textual satisfaction assessment [1]. The objective of the system is to perform the textual satisfaction method. Satisfaction assessment helps in identifying the unsatisfied requirements. The RTM for the data set is constructed and the requirement and design text into chunks. Stop word removal and the stemming for the chunks are performed. The chunks are tokenized into individual terms. The synonym pairs for the terms are determined. For TF-IDF and Naïve Satisfaction method, the threshold values are predefined. For NLP satisfaction method, the rules are generated. Finally the candidate satisfaction assessment mapping is performed.

Holbrook E A, Hayes J H and Dekhtyar A proposed automating requirements satisfaction assessment [2]. The system introduces the automation of satisfaction assessment, the process of determining the satisfaction mapping of natural language textual requirements to natural language design elements. The system describes the satisfaction assessment approach algorithmically and then evaluates the effectiveness of two proposed information retrieval (IR) methods in two industrial studies. Mainly focuses on assessing whether requirements have been satisfied by lower level artifacts such as design. A three-step approach to addressing the problem is proposed. The important components for each individual requirement/design element are identified. The satisfaction assessment as determination of whether each component of each requirement has been addressed in the design document is defined. The requirements document, broken into components to the design document, also broken into chunks is traced. Two methods are used to determine the mapping of requirement to design element chunks. The first method is based on a simple idea of tracking and thresholding the percentage of common terms between the two chunks. The second method is vector space information retrieval using TF-IDF term weighting.

Hayes J H, Dekhtyar A, Sundaram S, Holbrook A and Vadlamudi S proposed a tool for improving software maintenance through traceability recovery [3]. The recovery of traceability for artifacts containing unstructured textual narrative is addressed. RETRO uses information retrieval (IR) and text mining methods to construct candidate traces. The task is to find documents in the collection that are deemed relevant to the query. The tool consists of a set of IR and text mining methods as well as a front-end that provides functionality for the analyst to use during the tracing process. The method vector space retrieval with tf-idf term weighting is the default tracing technique in RETRO. Each document and query is passed through a stop word removal procedure. The remaining text is stemmed to ensure that words such as "information," "informational" and "informative" are treated as the same term. RETRO includes support for relevance feedback.

Jane Huffman Hayes, Alex Dekhtyar and Senthil Karthikeyan Sundaram proposed a new method for requirements tracing [4]. The issues related to improving the overall quality of the dynamic candidate link generation for the requirements tracing process for Verification and Validation are addressed. The goals for a tracing tool based on analyst responsibilities in the tracing process are defined. The several new measures for validating that the goals have been satisfied are introduced. The analyst feedback in the tracing process is implemented. A prototype tool, RETRO (REquirements TRacing On-target), to address the goals is presented.

The methods and tool can be used to trace any textual artifact to any other textual artifact. An additional IR technique, Latent Semantic Indexing is used for requirements tracing. A requirements tracing tool is defined that is a special purpose software that takes as input two or more documents in the project document hierarchy and outputs a traceability matrix that is a mapping between the requirements of the input documents. Two IR algorithms TF-IDF vector retrieval and vector retrieval with a simple thesaurus one newly implemented method, Latent Semantic Indexing are used for determining requirement similarity. LSI is a dimensionality-reduction method, which allows one to capture the similarity of underlying concepts, rather than simple keyword matches.

Robinson W N presented an implementation of rule-based monitors, which are derived from system requirements [5]. A language for requirements and monitor definitions are defined by the framework. A methodology for defining requirements, identifying potential requirements obstacles, and analyzing monitor feedback is defined. The framework address three interrelated monitoring issues such as Formalization of high-level goals, requirements, and their monitors, Automation of monitor generation, deployment, and optimization and Traceability between high-level descriptions and lower-level run-time events. The monitoring approach integrates requirements language research with commercial business process monitoring. The approach defines the logical monitoring model. The goals and requirements are defined. Potential requirements obstacles are uncovered and their monitors are derived. The monitoring architecture and implementation are defined. The requirements of the monitoring event sources and sinks are defined. A logical-physical mapping to ensure traceability of events back to requirements is defined. The monitoring system is implemented and deployed. The high-level feedback on the systems actions and requirements compliance is provided. The compensation and adaptation rules are executed when violations occur. The high-level feedback on the monitoring system itself, thereby providing historical information used in defining new monitoring optimization rules is provided.

Marcus A and Maletic J I proposed a method to recover traceability links between documentation and source code, using an information retrieval method, namely Latent Semantic Indexing [6]. The traceability links based on similarity measures are identified. The method utilizes all the comments and identifier names within the source code to produce semantic meaning with respect to the entire input document space. The vector space model (VSM) is a widely used classic method for constructing vector representations for documents. Latent Semantic Indexing (LSI) is a VSM based method for inducing and representing aspects of the meanings of words and passages reflective in their usage. LSI uses a user constructed corpus to create a term-by-document matrix. New document vectors (and query vectors) are obtained by orthogonally projecting the corresponding vectors in a VSM space (spanned by terms) onto the LSI subspace. The LSI subspace captures the most significant factors (i.e., those associated with the largest singular values) of a term-by-document matrix, it is expected to capture the relations of the most frequently co-occurring terms.

Cleland-Huang J, Chang C K, Sethi G, Javvaji K, Haijian H U and Jinchun Xia proposed a method for establishing and utilizing traceability links between requirements and performance models [7]. An activity that is of critical importance to handling and managing changing requirements effectively is described. A method for establishing and utilizing traceability links between requirements and performance models is proposed. Traceability links are established through the use of a dynamic traceability scheme capable of speculatively driving the impacted models whenever a quantitative requirement is changed. Key values from within the individual performance models representing probabilities, rates, counts and sizes etc are placed in the central requirements repository. Finely tuned links are then established between the data-values in the models and those in the repository. The process of analyzing the impact of a proposed change upon the performance of the system through dynamic re-execution of requirement dependent models is supported.

Giuliano Antoniol, Gerardo Canfora, Gerardo Casazza, Andrea De Lucia and Ettore Merlo proposed a method based on information retrieval to recover traceability links between source code and free text documents [8]. The method proposed ranks the free-text documents against queries constructed from the identifiers of source code components and can be customized to work with different IR models. Both a probabilistic and a vector space information retrieval model are applied. In the probabilistic model, free-text documents are ranked according to the probability of being relevant to a query computed on a statistical basis. A language model for each document or identifiable section is estimated and uses a Bayesian classifier to score the sequences of mnemonics extracted from each source code component against the models. The vector space model treats documents and queries as vectors in an n-dimensional space. Documents are ranked against queries by computing a distance function between the corresponding vectors. The documents are ranked according to a widely used distance function, i.e., the cosine of the angle between the vectors. The construction of the vocabulary and the indexing of the documents are preceded by a text normalization phase performed in three steps. In the first step, all capital letters are transformed into lower case letters. In the second step, stop-words (such as articles, punctuation, numbers, etc.) are removed. In the third step, a morphological analysis is used to convert plurals into singulars and to transform conjugated forms of verbs into infinitives. The construction of a query consists of three steps. Identifier extraction parses the source code component and extracts the list of its identifiers. Identifier separation splits identifiers composed of two or more words into separate words. Text normalization

applies the three steps described above for document indexing. Finally, a classifier computes the similarity between queries and documents and returns a ranked list of documents for each source code component.

III. Methodology

The system design is shown in Fig 2.

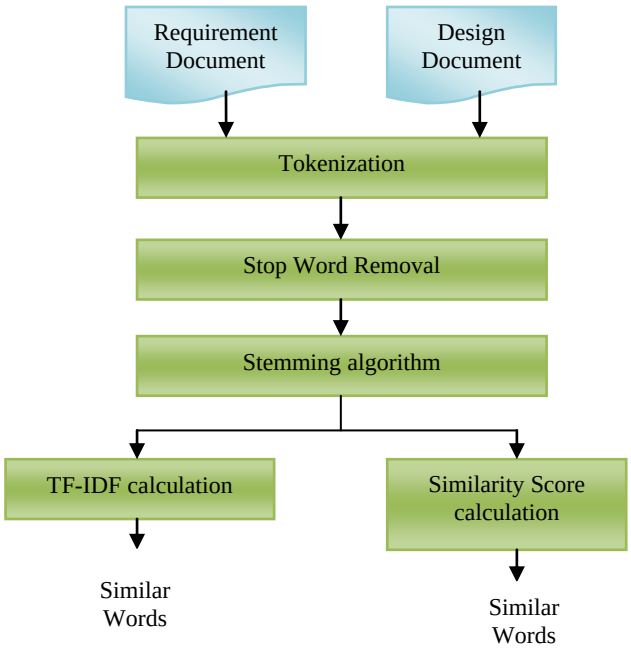

Fig. 2. System design

Fig 2 shows that the input is the requirement document and the design document that was written in natural language. It aims to determine the similarities between the requirement document and the design document. The sentences in the documents are tokenized into individual words. The stop word removal and stemming are performed. The term frequency for each word is calculated to determine the TF-IDF similarity calculation. The number of occurrence of the terms is determined to calculate the similarity score. Finally the similar words are calculated.

A. Document Preprocessing

The requirement document and the design document are given as the input. The preprocessing of the document is performed. Stop word removal find outs the words that have least importance in the phrases. Stemming is the process for reducing derived words to their stems. Porter's stemmer algorithm [12] is used to perform the stemming process. The Porter's stemmer algorithm consists of the following steps:

Step 1: Gets rid of plurals and -ed or -ing suffixes.

Step 2: Turns terminal y to i when there is another vowel in the stem.

Step 3: Maps double suffixes to single ones: -ization, -ational, etc.

Step 4: Deals with suffixes, -full, -ness etc.

Step 5: Takes off -ant, -ence, etc.

Step 6: Removes a final -e

B. Similarity score calculation

The similarity score of the document is calculated using the Naive satisfaction method. The similarity score of each of the term in both the requirement document and design document is determined using the following equation

$$sim = \frac{\text{Number of times the term occur}}{\text{total number of terms in the document}} \quad (1)$$

The similarity score of each word in the document is stored. The value of each of the term in the requirement document is compared with the term in the design document. The similar terms are determined by comparing the similarity score measure.

C. TF-IDF similarity calculation

The TF-IDF of each of the word in the document is calculated. Term frequency–inverse document frequency, is a numerical statistic that reflects how important a word is to a document in a collection or corpus. Term frequency is the count of number of times a particular word / term occurred in a document. Inverse document frequency, IDF is calculated using the following equation

$$\text{IDF} = \log\left(\frac{\text{N}}{\text{DF}}\right) \quad (2)$$

where, N is the total number of documents in a collection, and DF is the document frequency that is the number of documents where a given term occurs.

TF-IDF weight for each term is calculated using the following equation

$$\text{TF} - \text{IDF} = \text{TF} * \text{IDF} \quad (3)$$

The TF-IDF similarity measure of each word in the document is stored. The value of each of the term in the requirement document is compared with the term in the design document. The similar terms are determined by comparing the TF-IDF similarity measure.

D. Performance analysis

Precision and recall of both the methods are compared. Precision is the fraction of retrieved documents that are relevant to the find. Precision is calculated by using the following equation.

$$precison = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \quad (4)$$

Recall in information retrieval is the fraction of the documents that are relevant to the query that are successfully retrieved. Recall is calculated using the following equation

$$recall = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|} \quad (5)$$

IV. Results

Fig 3 shows the stemming of the terms in the documents using Porter's stemmer algorithm.

Fig 4 shows the calculation of the similarity score of each term in the document using equation (1).

Fig 5 shows the determination of the similar words based on the similarity scores of each of the term in the documents

Fig 6 shows the calculation of the TF-IDF value of each term in the document using equation (3).

Fig 7 shows the determination of the similar words based on the TF-IDF value of each of the term in the documents.

Table 1 shows the calculation of precision and recall of some of the elements in the requirement documents using naive method and TF-IDF method.

Table 2 shows the average precision and recall value of the naive and TF-IDF method. For the similarity score calculation method the precision and recall is 16% and 74%. For the TF-IDF value calculation method the precision and recall is 31% and 67%. The F-measure for the similarity score calculation method is 26% and that of TF-IDF value calculation method is 43%.

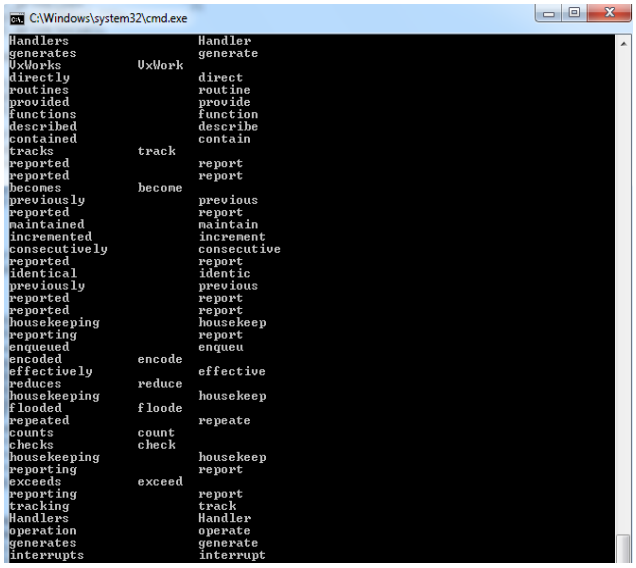

Fig. 3. Stemming of the terms in the document

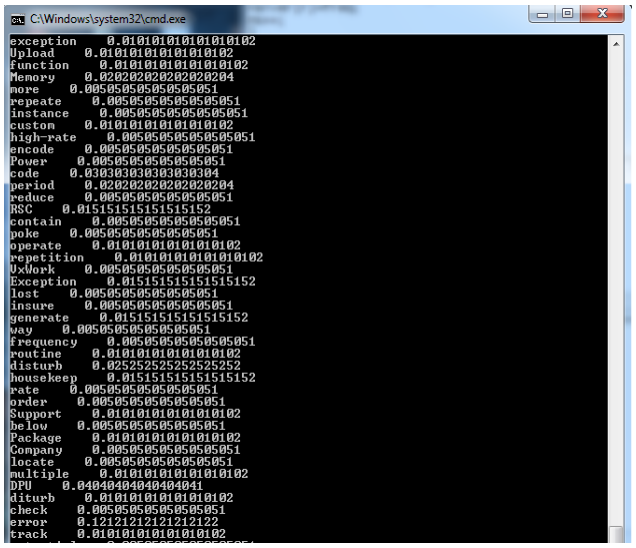

Fig. 4. Similarity score calculation

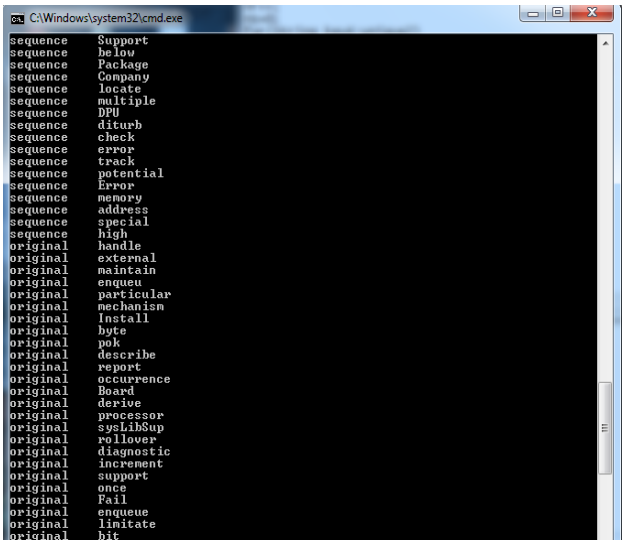

Fig. 5. Similar words determination using similarity score

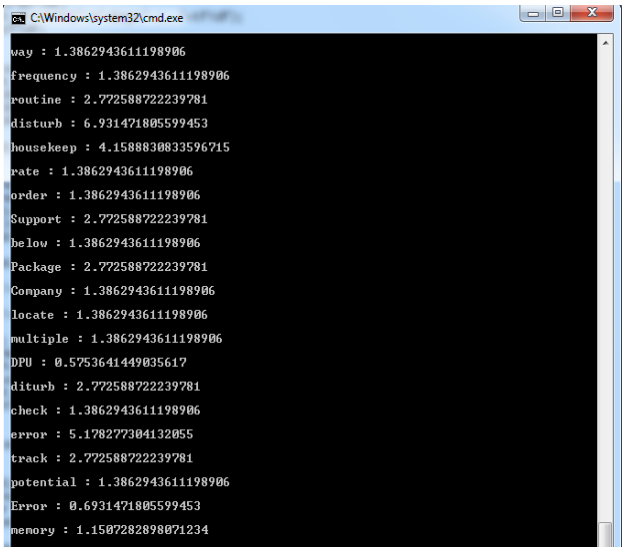

Fig. 6. TF-IDF calculation

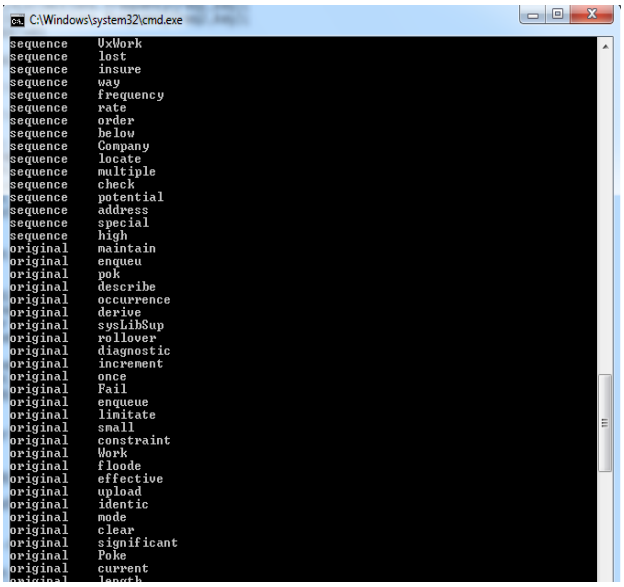

Fig. 7. Similar words determination using TF-IDF value

Table 1. Precision and Recall calculation

Elements	Naive method		TF-IDF method	
	Precision	Recall	Precision	Recall
Timeout	0.111	0.666	0.1935	0.568
Number	0.1759	0.84	0.30645	0.76
Complete	0.1388	1.058	0.2419	0.8823
Sequence	0.1666	0.68	0.3387	0.84
Original	0.1481	0.76	0.2903	0.72

Table 2. Average Precision and Recall

Method	Average Precision	Average Recall	F-Measure
Naive method	16%	74%	26%
TF-IDF method	31%	67%	43%

V. Conclusion

In this paper two methods for the satisfaction assessment of requirements are proposed. The methods are based on the TF-IDF value calculation and the similarity score calculation. The methods find out the satisfied requirements in the design document by identifying the similar words in both the documents. Porter's stemmer algorithm could be used to reduce the derive words into their stems. The two methods provide the terms that are similar in the requirement document and design document. The method based on TF-IDF value is more efficient compared to the method based on similarity score value based on their precision and recall value. The concept of contextual tracing could be included for the future work. Contextual tracing could help to find out the similar words that share the same meaning.

References

[1] Elizabeth Ashlee Holbrook, Jane Huffman Hayes, Alex Dekhtyar, Wenbin Li. A study of methods for textual satisfaction assessment. Springer- Empirical Software Engineering,2013,18(1):139-176.

[2] Holbrook Ea, Hayes J H, Dekhtyar A. Towards automating requirements satisfaction assessment. In: Proceedings of IEEE International Conference on Requirements Engineering, 2009, 149 – 158.

[3] Hayes J H, Dekhtyar A, Sundaram S, Holbrook A, Vadlamudi S. Requirements Tracing on Target (RETRO): Improving software maintenance through traceability recovery. Springer Innovations System Software Engineering,2007,3(3):193-202.

[4] Jane Huffman Hayes, Alex Dekhtyar, Senthil Karthikeyan Sundaram. Advancing candidate link generation for requirements tracing: the study of methods. IEEE Transactions on Software Engineering,2006,32(1): 4 – 19.

[5] Robinson W N. Implementing rule-based monitors within a framework for continuous requirements monitoring. In: Proceedings of Annual Hawaii International Conference on System Sciences, 2005, 188a.

[6] Marcus A, Maletic J I. Recovering documentation-to-source code traceability links using latent semantic indexing. In: Proceedings of International Conference on Software Engineering, 2003,125-135.

[7] Cleland-Huang J, Chang C K, Sethi G, Javvaji K, Haijian H U, Jinchun Xia. Automating speculative queries through event-based requirements traceability. In: Proceedings of IEEE Joint Conference on Requirements Engineering,2002,289-296.

[8] Giuliano Antoniol, Gerardo Canfora, Gerardo Casazza, Andrea De Lucia, Ettore Merlo. Recovering traceability links between code and documentation. IEEE Transactions On Software Engineering,2002,28(10): 970 – 983.

[9] Roger S Pressman. Software Engineering: a practitioner's approach. 6th edition, McGraw-Hill Pub Co, New York,2005.

[10] Phillip A Laplante. Requirements Engineering for Software and Systems. 2nd edition, CRC press, New York.

[11] Donald Firesmith. Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them. Journal of Object Technology,2007,6(1).

[12] Noraida Haji Ali, Noor Syakirah Ibrahim. Porter Stemming Algorithm for Semantic Checking. In: Proceedings of 16th International Conference on Computer and Information Technology, 2012, 253 – 258.

Authors' Profiles

Ms. **Divya K.S**. was born in Ernakulam, India in 1990. She received Bachelor of Technology degree in Computer Science and Engineering under Cochin University, in 2012. She is currently pursuing Master of Engineering degree in Computer Science and Engineering under Anna University, Chennai, India.

Dr.R.Subha received B.E in Computer Science and Engineering from Periyar University and M.E in Software Engineering from Anna University, Chennai in 2002 and 2006 respectively and completed the Ph.D. degree in software engineering in 2014. At Present, she is working as Assistant Professor in the department of Computer Science & Engg, Sri Krishna College of Technology, Coimbatore. She is currently pursuing Ph.D under Anna University, Coimbatore. Her research interest includes Software Engineering, Computer Architecture and NLP.

Dr.S.Palaniswami received the B.E. degree in electrical and electronics engineering from the Govt., college of Technology, Coimbatore, University of Madras, Madras, India, in 1981, the M.E. degree in electronics and communication engineering (Applied Electronics) from the Govt., college of Technology, Bharathiar University, Coimbatore, India, in 1986 and the Ph.D. degree in electrical engineering from the PSG Technology, Bharathiar University, Coimbatore, India, in 2003. He is currently the Registrar of Anna University Coimbatore, Coimbatore, India, Since May 2007. His research interests include Control systems, Communication and Networks, Fuzzy logic and Networks, AI, Sensor Networks. He has about 25 years of teaching experience, since 1982. He has served as lecturer, Associate Professor, Professor, Registrar and the life Member of ISTE, India.

