ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

Advanced Matching Technique for Trustrace To Improve The Accuracy Of Requirement

S.Muthamizharasi[1], J.Selvakumar[2], M.Rajaram[3]
PG Scholar, Dept of CSE (PG)-ME (Software Engineering), Sri Ramakrishna Engineering College, Coimbatore, India[1]
Professor, Dept of CSE (PG)-ME (Software Engineering), Sri Ramakrishna Engineering College, Coimbatore, India[2]
Professor, Anna University, Chennai, India[3]

Abstract: During software evolution and maintenance, requirement traceability links become outdated because developers could not dedicate effort to updating them. So far, recovering these traceability links later is intimidated and costly task for developers. Earlier methods used information retrieval for recovering traceability links between free-text requirements and source code. However the accuracy is limited in such methods. To deal with this, the proposed system uses Relink, an automatic link recovery algorithm. The proposed approach is based on automatically learned feature criteria from explicit links. In addition the missing links can be found which affects defect prediction performance. We also assessed the impact of recovered links on software maintainability measurement, and established the result of ReLink affords considerably better accuracy than those of conventional heuristics.

Keywords: Traceability, Link, Relink, Histrace, Trumo, DynWing

I. INTRODUCTION

Requirements traceability has obtained much consideration over the past decade in the scientific literature. It is defined as “the ability to illustrate and go after the life of a requirement, in both a forward and backward direction”. Traceability links among the requirements of a system and its source code aids in reducing system comprehension attempt. Until now, during maintaining and evolving the Software, the developers can add, remove, or modify features. Requirement traceability links turn into obsolete because developers cannot devote effort to updating them. However recovering these traceability links later is a daunting task for developers and also they frequently evolve requirements and source code in a different way. In fact, they usually do not update requirement-traceability links with source code. Requirements and source code subsequently different from each other, which decreases the textual similarity.

Trustrace is defined as a traceability-recovery approach between the requirements and source code, which is mainly used to improve the accuracy of Traceability. To dynamically discard or rerank the traceability links reported by an IR technique it uses heterogeneous sources of information. Trustrace consists of three parts:

- Histrace
- Trumo
- DynWing

Histrace

To create links between requirements and source code Histrace mines software repositories by using information from the repositories. It stores all the recovered links between requirements and software repositories in sets.e.g., Histracecommits and Histracebugs are considered as experts whose opinions will be used to discard or rerank baseline traceability links.

Trumo

Trumo merges the requirement traceability links attained from an IR technique and discards or reranks them using an expert’s opinions and a trust model motivated by web-trust models. It compares the similarity of the recovered links with the result provided by the experts and with the number of times the link appears in each expert’s set. This is not fixed to any definite IR-based traceability-recovery approach and thus it can use any expert’s view to correct the ranking of recovered links.

ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

DynWing

DynWing calculates and allocates weights to the experts in the trust model dynamically. Thus, dynamic-weighting techniques are promoted to assign weights for each link. DynWing initializes each expert's similarity value for each link and assigns weights depends on to these values.

To discover possibilities of recovering the missing links automatically, a qualitative study has been conducted to recognize characteristics of explicit links based on the bug IDs in change logs. The links between bugs and changes has been found and demonstrated certain features. For instance, the bug-fixing time is relatively close to the change logs, change-commit time and the bug reports share textual similarity, and the developers accountable for a bug are usually the committers of the bug-fixing change. According to these findings, we propose a ReLink, an automatic link recovery algorithm. Relink automatically learns the criteria of features from explicit links, and applies the learned criteria which checks whether the features of an unknown link assure the criteria. The valid link can be obtained when the unknown link satisfies all the feature criteria.

II. RELATED WORK

The literature used many methods, techniques, and tools to recover the traceability links semiautomatically or automatically.

Requirements traceability has obtained much notice over the past decade in the scientific literature. To recover traceability links between high-level documents various researchers used information retrieval (IR) techniques, e.g., [1], [2], [3].

S.A. Sherba and K.M. Anderson in [4] presented a new approach to traceability based on practices from information integration and open hypermedia. Information integration and Open hypermedia offers generic techniques for establishing, viewing and maintaining connections between artifacts of software. This approach permits the automated creation, viewing and maintenance of traceability relationships in tools in which software professionals are get to know using on a daily basis. Though, due to the heterogeneous nature of this artifact, creating, viewing and maintaining these relationships is intensely difficult.

Maider et al.in [5] presented an approach to hold automated traceability maintenance by recognizing development behavior. Development behavior is officially specified and changed to definite model elements triggered a LinkUpdateManager. LinkUpdateManager is accountable for updating traceability links that are shared to the changed elements. Though, the authors did not declare how traceability links are in fact created and updated.

Lucia et al.in [6] presented an approach supporting developers to keep source code identifiers and comments reliable with high-level artifacts. This approach calculates textual similarity between related high-level artifacts and source code. The textual similarity supports developers to develop their source code list.

Maletic and Collard in [7] presented TQL, an XML-based traceability query language. TQL helps the queries among multiple artifacts and traceability link types. TQL has primitives to permit complex query construction and execution support. In this the following question arised that how does a change impact the requirements? How does a change to the requirements impact the safety of the system?

Gethers in [8] exploited the empirical finding and presented an integrated approach to combine orthogonal IR techniques. Thus this has been statistically shown to create dissimilar results. The presented approach merges the following IR-based methods: probabilistic Jensen and Shannon (JS) model, and Relational Topic Modeling (RTM) and Vector Space Model (VSM), which has not been used in the framework of traceability link recovery prior. However this returns that the satisfaction measurement of user requirements give less result..

III. EXISTING SYSTEM

In existing system Histrace creates links between the set of requirements and source code using the software repositories. It considers the requirements textual descriptions, CVS/SVN commit messages, classes and bug reports as separate documents.

These sources of information were used to produce two experts, which we call Histracecommits and Histracebugs, which use CVS/SVN commit messages and bug reports respectively. Histrace must link CVS/SVN commit messages to bug reports before being able to exploit bug reports for traceability.

In this Histrace uses regular expressions that are a simple text matching approach except with reasonable results, to link CVS/SVN commit messages to bug reports.

As a result, Histrace takes that developers allocated to each bug which has a unique ID that is a sequence of digits recognizable through regular expressions. The same ID must be preferred to by developers in the CVS/SVN commit messages.

To link CVS/SVN commit messages to bug reports, Histrace performs the following steps:

1) All CVS/SVN commit messages are extracted along with commit status and committed files.

2) All the bug reports are being extracted, along with textual descriptions and time/date and

3) Each CVS/SVN commit message and bug reports are linked by using regular expressions, e.g.

((B)[UG]{0,2}\S*[ID]{0,3}|ID|FIX|PR|#)[\S#=]*[?(0—9]{4,6})]?

Which is the regular expression refrained to the numbering and naming conventions used by the developers. This expression to be updated which matches with other numbering and naming conventions.

At last Histrace takes out false-positive links by imposing the following constraint

Fix(e[ds])?|bugs?|problems?|defects?patch".

The above regular-expression constraint keeps only a CVS/SVN commit when it contains a keyword. Thus it returns the bug reports linked to CVS/SVN commit messages.

IV. PROPOSED SYSTEM

In proposed CVS/SVN commit message and bug reports are linked using the ReLink method, this method recovers the link between bug and changes.

A.Features of Links

The proposed method finds the features of links between bugs and changes. The following features identify and recover the missing links.

Time Interval: This is the interval between the time (tf) when a bug was fixed as given by its bug report and the time (tc) when the corresponding bug fix was committed at the code repository.This feature is useful because it fix the bugs after its creation and before closure.

Bug Owner and Change Committer

For linked bugs and changes, there exists a mapping between the bug owner and the change committer.

Committer: the person who is responsible for fixing the bug .The mapping between these two persons could be identified through mining of software repositories.

Text Similarity

This is the textual similarity between bug reports and change logs. For the concept of linked bugs and changes, the natural language descriptions in the bug report are frequently analogous to those in the change logs, as they may points to the same issue and share similar keywords.

B.Link feature mining

The Interval between the Bug-fixing time and the Change-Commit Time: Even though the interval between commit time and bug-fixing time is a useful feature for mining links, identifying the length of such an interval is a difficult task. It is observed from evaluation that developers do not often change the status of bugs to "Fixed" in the bug tracking system abruptly after they have committed the changes of bug-fixing. The change-commit time and the bug-fixing time could be far apart. Further investigations found that most bug comments are close to bug-fixing activities, while developers frequently post comments to the bug tracking system to report a bug fix and inform the bug reporter. Consequently, the bug-fixing time according to the time of comments in bug reports were also to be determined.

Mapping between Change Committers and Bug Owners

To identify the mappings between bug owners and change committers, the comments in bug reports were examined. These empirical studies found that developers frequently discuss bug-related issues and declare bug fixes through the bug tracking system. As a result, it is likely that one of the commenters is the bug owner who is liable for bug fixing.

The Similarity between Bug Reports and Change Logs

ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

In this project, bug reports and change logs are treated as texts and their similarities are compared. It is anticipated that for linked bugs and changes, change logs reveal certain similarity. In order to compute the similarity between bug reports and change logs, text features are need to be extracted.

To select the delegate terms in the text, the following steps to be used to reduce dimensions for long bug reports and change logs.

1. Remove stop words
2. Use one term to represent all other terms that have the same stemmer.
3. Use one term to represent all synonymous words

C. Features Criteria Learning

In this the thresholds of features can be described so that these features can distinguish most of the real links.

The following algorithm 1 determines the threshold value of features

```
Algorithm 1: Threshold determination
        DetermineThresholds (Le: links between bugs
and changes identified by the traditional heuristics)
1 Assign the time interval T with a small initial value T0
2 Assign the text similarity threshold S with a small initial
value S0
3 Select links in Le that satisfy T and S, and compute F-
measure
4 Increase S by a small step s1
5 Repeat steps 3-4 until the maximum threshold Sm is
reached
6 Increase T by a small step s2
7 Repeat steps 3-6 until T reaches the maximum threshold
Tm
8 Choose the threshold values Tt and St that achieve the
best F-measure
9 Return Tt and St.
```

The mappings between bug owners and change committers were also learnt from the explicit links which is identified by the conventional heuristics. These mappings can be determined by using following algorithm2:

```
Algorithm 2:Mapping Determination
        DetermineMappings (Le: links between bugs and
changes identified by the traditional heuristics)
1 Initialize the set of mappings M=Φ
2 For each link l in Le
3 For each mapping m between l's change committer
and bug commenter
4 If (m is not in M)
5 Add m to M
6 EndIf
7 EndFor
8 EndFor
9 Return M
```

D.Recovering the Missing Links

To automatically identify the links between bugs and changes, a ReLink has been proposed which is a link recovery approach. ReLink is based on the identified features. The algorithm 3 of ReLink is described below:

```
Algorithm3: Relink approach
Store all possible links between bugs and links in L
2 Initialize the set Lr =Φ
3 Mine links Le between bugs and changes using
the traditional heuristics
4 DetermineThresholds (Le)
5 DetermineMappings(Le)
6 For each link l in (L – Le)
7 If there is mapping between l's bug commenter and l's
change committer
8 If any of l's bug comment time is within the time
interval threshold Tt
9 If the text similarity between l's bug report and
change log is within threshold St
10 add l to Lr
11 EndIf
12 EndIf
13 EndIf
14 EndFor
15 Return Lr + Le
```

V. EXPERIMENTAL RESULT

The proposed system can be evaluated based on traceability links such as baseline requirements, JSM and VSM. The qualitative analyses of the proposed system results and discuss observations from the empirical evaluation of Trustrace.

Data Set Quality Analysis

The empirical evaluation of proposed system supports the Trustrace combined with IR techniques is effective in increasing the precision and recall values of some baseline requirements traceability links. This uses both JSM and VSM to recover traceability links and compare their results in isolation with those of Trustrace. The proposed system performs better than existing approach in terms of precision, recall and f-measure.

The following comparison table1 provide the estimated values for the existing and proposed approach.

ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

TABLE 1

COMPARISION TABLE

Metrics	Existing system			Proposed system		
	Base line IR	JSM	VS M	Base line IR	JSM	VS M
Precision	59.5	73.14	78.5	61.2	77.143	86.1
Recall	65.1	78.52	83.5	66.3	78.92	85
Fmeasure	61.3	75.74	81.3	63.1	78.025	84.5

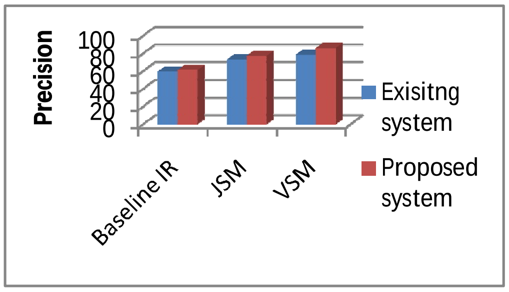

Fig.1 Precision comparison

The above graph in figure 1 shows precision rate in which the proposed system performs higher than existing in the traceable links of Baseline IR, JSM and VSM respectively.

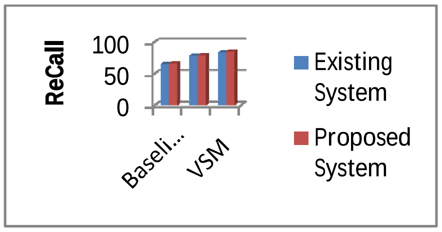

Fig. 2 Recall comparision

The above graph in figure 2 shows recall rate in which the proposed system performs higher than existing in the traceable links of Baseline IR, JSM and VSM respectively.

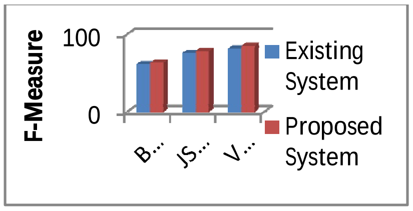

Fig. 3 F-measure comparison

The above graph in figure 3 shows f-measure rate in which the proposed system performs higher than existing in the traceable links of Baseline IR, JSM and VSM respectively.

VI. CONCLUSION

In this paper, a traceability-recovery approach between requirements and source code is supported on three conjectures: Histrace, Trumo, and DynWing. The proposed Relink recovers the link between bug and changes in Histrace which creates links between the set of requirements and source code using the software repositories .The features of links are identified and mined in the above discussion. Finally missing link can be learned by using Relink method followed by determining the Threshold and mapping criteria's. Experimental result for various traceability links has been demonstrated .In that proposed system outperforms better than existing system.

REFERENCES

[1]. 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.(2)

[2]. 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.(3)

[3]. Marcus and J.I. Maletic, "Recovering Documentation-to- Source-Code Traceability Links Using Latent Semantic Indexing," Proc. 25th Int'l Conf. Software Eng., pp. 125-135, 2003(4)

[4]. S.A. Sherba and K.M. Anderson, "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,(30)

[5]. P. Mader, O. Gotel, and I. Philippow, "Enabling Automated Traceability Maintenance by Recognizing Development Activities Applied to Models," Proc. 23rd IEEE/ACM Int'l Conf. Automated Software Eng., pp. 49-58, 2008.(31)

[6]. A.D. Lucia, M.D. Penta, and R. Oliveto, "Improving Source Code Lexicon via Traceability and Information Retrieval," IEEE Trans. Software Eng., vol. 37, no. 2, pp. 205-227, Mar. 2011.(32)

[7]. J.I. Maletic and M.L. Collard, "TQL: A Query Language to Support Traceability," Proc. ICSE Workshop Traceability in Emerging Forms of Software Eng., pp. 16-20, 2009.(6)

[8]. 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 Maintenance, pp. 133-142, Sept. 2011.(9)