Recovering Transitive Traceability Links among Software Artifacts

Kazuki Nishikawa, Hironori Washizaki
Yoshiaki Fukazawa
dept. Computer Science
Waseda University
Tokyo, Japan
t260g.v-max.7@ruri.waseda.jp

Keishi Ohshima, Ryota Mibe
Yokohama Research Laboratory
Hitachi, Ltd
Kanagawa, Japan
keishi.oshima.rj@hitachi.com

***Abstract*—Although many methods have been suggested to automatically recover traceability links in software development, they do not cover all link combinations (e.g., links between the source code and test cases) because specific documents or artifact features (e.g., log documents and structures of source code) are used. In this paper, we propose a method called the Connecting Links Method (CLM) to recover transitive traceability links between two artifacts using a third artifact. Because CLM uses a different artifact as a document, it can be applied to kinds of various data. Basically, CLM recovers traceability links using the Vector Space Model (VSM) in Information Retrieval (IR) methods. For example, by connecting links between A and B and between B and C, CLM retrieves the link between A and C transitively. In this way, CLM can recover transitive traceability links when a suggested method cannot. Here we demonstrate that CLM can effectively recover links that VSM cannot using Open Source Software.**

Index Terms—traceability link recovery, transitive traceability links, connecting links

I. INTRODUCTION

Traceability indicates that the relationship between two objects can be traced. These relationships are known as traceability links, and are used in various situations. In software development, traceability links are used to understand the relationships between software artifacts (e.g., requirements, designs, source code and test cases), helping developers discover demand and implementation errors. If requirements are changed and the source code must be modified, the source code can easily be rewritten if the relationship between the requirements and the source code is known. Hence, traceability leads to a reduction in development costs [4, 18, 19].

Previous methods to automatically recover traceability have limited applications such as recovering traceability links between specific artifacts [3, 7, 8, 13, 17, 20, 21, 22] because they use specific documents or artifact features (e.g., log documents or structures of source code). Herein we propose a method, which is called the Connecting Links Method (CLM), to recover transitive traceability links between two artifacts using a third artifact (e.g., requirements, designs, source code, and test cases). We call our proposed method the Connecting Links Method (CLM). For example, by connecting links between A and B and between B and C, CLM retrieves the link between A and C transitively. These artifacts are found in almost all software developments. Although it is uncertain whether specific documents actually exist in software development, if a different artifact is used, then this is not an issue, which is why we assume that CLM is superior.

This paper aims to address the following Research Questions:

RQ1 Is VSM limited on the kinds of data it can recover?
RQ2 Can CLM recover links that VSM does not?
RQ3 How much does the F-measure of CLM improve compared to VSM?

This paper makes the following contributions.

- We propose a method, CLM, to automatically recover traceability links using a different artifact.
- We demonstrate a situation in which CLM applies.
- We confirm the effectiveness of CLM under the appropriate conditions.

This paper is organized as follows. Section 2 describes background information about CLM. Information about our approach is presented in section 3. Section 4 evaluates CLM by conducting experiments on a target. Section 5 discusses related works. Finally, the conclusion and future works are provided in section 6.

II. BACKGROUND

A. VECTOR SPACE MODEL (VSM)

VSM [11] is an Information Retrieval (IR) [2, 10, 12] method. IR methods research objective information from vast amounts of information, and are typically used for site searches on the Internet to find specific information. IR methods can also recover traceability links. For example, VSM can find similar documents, which are vectors in the space of terms, by using their cosine distances to compute their textual similarity. Then the documents are ranked by these similarities. There are many IR methods [23]. However, many of these are difficult to use because variables must be inputted. In contrast, VSM does not require such variables. Consequently, CLM employs VSM.

B. MOTIVATING EXAMPLE

Figure 1 describes the data of CC_1, ID_1 and TC_1[1] in EasyClinic, which is a software program. UC, ID, CC, and TC

[1] These data numbers differ from real data.

indicate use cases, interaction diagrams, code classes, and test cases, respectively. The link between CC_1 and TC_1 is a correct link, but it is hard to recover using suggested methods (e.g., log documents, or the source code structure) because the link does not have a log document or source code details. Although VSM can provide the link between CC_1 and TC_1, the precision is low because there is not a strong relation word between CC_1 and TC_1. This issue has motivated us to expand the coverage using a method that employs a third software artifact.

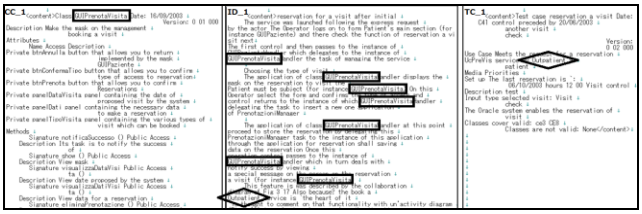

Fig. 1. Data of CC_1, ID_1 and TC_1

III. OUR APPROACH

A. Overview of CLM

Figure 2 overviews CLM. Artifact is a set of software artifacts (e.g., requirements, designs, source code, and test cases). In case of requirements, R indicates set of requirements and r_1 indicates an element of requirements. This is shown $R=\{r_1, r_2, \cdots, r_h\}$. So Artifact A, B and C are defined $A=\{a_1, a_2, \cdots, a_l\}$, $B=\{b_1, b_2, \cdots, b_m\}$ and artifact $C=\{c_1, c_2, \cdots, c_n\}$. Unlike VSM, which directly recovers traceability links between A and C, CLM initially recovers the traceability links between A and B and between B and C to determine the link between A and C. Then these traceability links are connected to determine desired link between A and C. Consequently, by connecting links, CLM can recover links that VSM misses and indirectly determine traceability links.

For example, in EasyClinic, CLM uses ID_1 to recover the link between CC_1 and TC_1 because the relationship between CC_1 and TC_1 is weak. In contrast, because the word "GUIPrenotaVisita" in CC_1 and the word "Outpatient" in TC_1 are included in ID_1, the relationships between CC_1 and ID_1 and between ID_1 and TC_1 are strong. Consequently, the link between CC_1 and TC_1 is found by connecting the link between CC_1 and ID_1 with the link between ID_1 and TC_1. In this way, CLM can recover missing links.

B. Process of CLM

Figure 3 depicts the CLM process. Artifact $X=\{x_1, x_2, \cdots, x_p\}$, $Y=\{y_1, y_2, \cdots, y_q\}$ and Document Artifact $Z=\{z_1, z_2, \cdots, z_r\}$ are software artifacts. For explanation, we denote a link between two elements e_i and e_j as *link_*e_ie_j, such as *link_*x_1y_1. To determine the traceability links between X and Y, the following steps are implemented:

Step 1: Recover the traceability links between X and Z and between Y and Z in VSM.

Step 2: Connect these two retrieved traceability links to recover the traceability links between X and Y.

Step 1 uses TraceLab, which has been employed in various projects [1, 6], to recover the traceability links in VSM. TraceLab is an automatic traceability link recovery tool [9, 14] by CoEST [24], an international organization founded in 2006 to tackle pervasive challenges of implementing effective software and systems level traceability. By using TraceLab, we recover the traceability links between X and Z and between Y and Z in VSM. At the same time, we define the score of links between X and Z and between Y and Z. The score, which is calculated in VSM, is the value of link relationship. The max value is 1.0, which means two artifacts are same. In contrast, the minimum value is 0.0, which means no relationship.

Step 2 connects the links recovered in step 1. We define the link between X and Y that relate to same Z. The score of this link is calculated by multiplying the score of the link between X and Z by the score of the link between Y and Z.

Links between X and Z and between Y and Z are ranked by score. At first, we select the top link (e.g. *link_*x_iz_j) between X and Z. Next, we check links between Y and Z from top to bottom. Then, if we find links (e.g. *link_*y_kz_j) relate to same z (e.g. z_j), links (e.g. *link_*x_iz_j and *link_*y_kz_j) are connected and search is stopped. This flow is repeated using from the next link to last link between X and Z. Because lower ranking results tend to contain incorrect links, only the top 10% of the rankings by score for each retrieved traceability link are evaluated in CLM. If multiple variables z_f and z_g are connected to both x_i and y_k, then the highest link score of links with z_f and z_g is selected. If there are multiple links between x_i and z_f and between x_j ($i \neq j$) and z_f related to one link between y_k and z_f, then y_k is connected to x_i and x_j.

For example, consider requirement R1, source code S1, and source code S2. If a link between R1 and design D1 has a score of 0.2 and a link between S1 and D1 has a score of 0.25, then R1 and S1 can be linked through D1. The score of R1 and S1 is 0.05 (0.2x0.25). However, if there are links between R1 and design D2 and between S1 and D2, which have scores of 0.4 and 0.25, respectively, the score of the link connected by D2 is 0.1 (0.4x0.25). In this case, the score of link between R1 and S1 is not 0.05 but 0.1. In addition, if there is a link between source code S2 and design D1, which has a score of 0.1, the link between R1 and S2 can also be determined, and the score of this link is 0.02 (0.2x0.1).

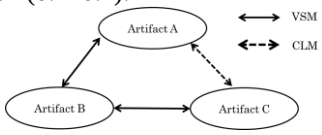

Fig. 2. Overview of CLM

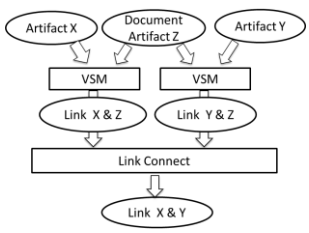

Fig. 3. Process of CLM

IV. Evaluation

To answer the three abovementioned research questions, we conducted a case study and evaluated CLM.

A. Case Study

EasyClinic, a software system to manage a medical doctor's office, was used as the data source. It contains four software artifacts: requirements, designs, source code, and test cases. These data have correct links between artifacts.

B. Experiments

1) Result of VSM (RQ1)

Table 1 compares the links between various artifacts. It should be noted that the listed F-measure is the maximum value. UC-ID denotes the traceability links between UC and ID. The max F-measures of ID-TC and ID-CC are 0.69 and 0.62, respectively, which are high values. On the other hand, the max F-measure of CC-TC is 0.45, which is low.

We also researched the documents for these links. The data of ID_1 and TC_1 are connected by the word "Outpatient". In the 20 files of ID, this word appears 3 times, and in the 63 files of TC, it appears 19 times. After calculating in IDF, the importance of "Outpatient" in ID and TC is 1.82 and 1.52.

The data of ID_1 and CC_1 are connected by the word "GUIPrenotaVisita". In the 20 files of ID, this word appears thrice, and in the 47 files of CC, it appears twice. After calculating in IDF, the importance of "GUIPrenotaVisita" in ID and CC is 1.82 and 2.37.

The data of TC_1 and CC_1 are connected by the words "visit" and "reservation". In the 63 files of TC, both words appear 26 times. In the 47 files of CC, "visit" appears 21 times and "reservation" appears 15 times. After calculating in IDF, the importance of "visit" and "reservation" in TC are both 1.38 and the importance of "visit" and "reservation" in CC are 1.34 and 1.50, respectively.

TABLE I. Comparison of Links between Various Artifacts by the Max F-measure

	Correct links	All links	Precision	Recall	F-Measure
UC-ID	26	600	0.362	0.808	0.5
UC-CC	93	1410	0.611	0.591	0.6
UC-TC	63	1890	0.466	0.54	0.5
ID-CC	69	940	0.661	0.594	0.62
ID-TC	83	1260	0.797	0.614	0.69
CC-TC	204	2961	0.4	0.52	0.45

2) Comparison of VSM and CLM (RQ2, RQ3)

Table 2 compares VSM and CLM. Correct links indicate the number of correct links in each traceability link. All links denote the total number of links restored by each method. Because CLM connects incorrect links, but with a low rank, we chose the top 20 precision and recall values in the max F-measure of the restored links. UC-ID(CC) means the traceability links between UC and ID in VSM and the traceability links between UC and ID using document CC in CLM. We selected cases of CC-TC(ID) and CC-TC(UC). The precisions of VSM and CLM are 0.55 and 0.95 for CC-TC(ID), respectively. The precisions of VSM and CLM are 0.55 and 0.35 in CC-TC(UC), respectively. These data are examples of high and low precision using CLM for the same traceability links such as CC-TC.

Figures 4 – 6 shows the precision, recall, and F-measure, respectively, for CC-TC(ID) and CC-TC(UC). The vertical axis denotes the index of the retrieved links, while the horizontal axis indicates the number of retrieved links. CLM is more effective than VSM for CC-TC(ID), but is ineffective for CC-TC(UC).

Table 3 provides differences in the top 20 links between VSM and CLM of CC-TC(ID) and CC-TC(UC). Correct links and wrong links are the number of correct links and incorrect links in the top 20 links recovered by VSM and CLM, respectively. Unique links and common links denote the number of unique links and common links, respectively, in the correct links, wrong links, and all links. The number of unique and correct links of CC-TC(ID) are 7 (15) in VSM (CLM). The number of common links in all links of CC-TC(ID), where CLM has a high effect, is 4. In contrast, the number of common links in all links of CC-TC(UC), where CLM has a low effect, is 8. In CC-TC(ID), the words used to connect links for CC-ID and ID-TC are "GUIPrenotaVisita" and "Outpatient", whereas while in CC-TC(UC) the same word "patient" is used in CC-TC(UC) to connect links for CC-UC and UC-TC.

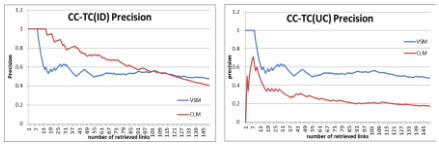

Fig. 4. Precision results of CC-TC(ID) and CC-TC(UC)

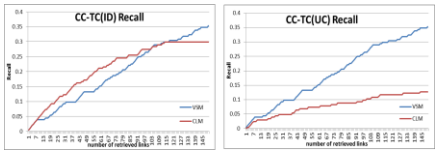

Fig. 5. Recall results of CC-TC(ID) and CC-TC(UC)

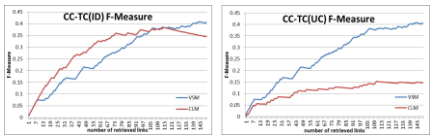

Fig. 6. F-Measure results of CC-TC(ID) and CC-TC(UC)

C. Discussion

RQ1 Is VSM limited on the kinds of data it can recover?

The high F-measures are 0.69 and 0.62 for the ID-TC and ID-CC data, respectively, while that for the CC-TC is low at 0.45. The values of IDF for the word "Outpatient" used to connect the ID-TC links are 1.82 in ID and 1.52 in TC. The values of IDF of the word "GUIPrenotaVisita" used to connect ID-CC links are 1.82 in ID and 2.37 in CC. The values of IDF of the words "visit" and "reservation" used to connect the TC-CC links are respectively 1.38 and 1.38 in ID and 1.34 and 1.50 in CC.

These results indicate that links using highly important words by VSM yield high F-measures. Words with high importance tend to be proper nouns, such as file names and class names. VSM can recover a lot of correct links with words of high importance, but not with words of low importance. Traceability links without a highly important word yield low F-measures using VSM. These results indicate that VSM is unsuited to recover links without highly important words.

RQ2 Can CLM recover links that VSM does not?

The precision of VSM (CLM) is 0.55 (0.95) for the top 20 links of CC-TC(ID), but is 0.55 (0.35) for the top 20 links of CC-TC(UC), demonstrating that CLM is more (less) effective than VSM for CC-TC(ID) (CC-TC(UC)). The number of unique and correct links of CC-TC(ID) is 7 (15) in VSM (CLM). The number of common links in all links of CC-TC(ID) is 4, whereas that of CC-TC(UC) is 8.

CLM recovers links that are not recovered by VSM in CC-TC(ID), but not in CC-TC(UC). We conjecture that the difference in the effectiveness in CLM is related to the connected links. In CC-TC(ID), because the words used to connect links CC-ID and ID-TC are "GUIPrenotaVisita" and "Outpatient", there are few common links. However, in CC-TC(UC) both use the word "patient" to connect links CC-UC and UC-TC, resulting in many common links. Consequently, CLM recovers different links than VSM when CLM determines links in a different manner than VSM.

TABLE II. COMPARISON BETWEEN VSM AND CLM IN THE MAX F-MEASURE

		VSM			CLM		
	Correct links	***All links***	***Precision***	***Recall***	***All links***	***Precision***	***Recall***
UC-ID(CC)	26	600	0.4	0.308	177	0.4	0.308
UC-ID(TC)	26	600	0.4	0.308	107	0.45	0.346
UC-CC(ID)	93	1410	0.85	0.183	204	0.75	0.161
UC-CC(TC)	93	1410	0.85	0.183	263	0.3	0.065
UC-TC(ID)	63	1890	0.55	0.175	274	0.15	0.048
UC-TC(CC)	63	1890	0.55	0.175	618	0.55	0.175
ID-CC(UC)	69	940	0.9	0.261	160	0.75	0.217
ID-CC(TC)	69	940	0.9	0.261	185	0.4	0.116
ID-TC(UC)	83	1260	0.8	0.193	184	0.65	0.157
ID-TC(CC)	83	1260	0.8	0.193	504	0.95	0.229
CC-TC(UC)	204	2961	0.55	0.054	542	0.35	0.034
CC-TC(ID)	204	2961	0.55	0.054	375	0.95	0.093

TABLE III. DIFFERENCES IN THE TOP 20 LINKS BETWEEN VSM AND CLM OF CC-TC(ID) AND CC-TC(UC)

				Correct		Wrong		All	
		Correct links	***Wrong links***	***Unique links***	***Common links***	***Unique links***	***Common links***	***Unique links***	***Common links***
CC-TC(ID)	VSM	11	9	7	4	9	0	16	4
	CLM	19	1	15	4	1	0	16	4
CC-TC(UC)	VSM	11	9	7	4	5	4	12	8
	CLM	7	13	3	4	9	4	12	8

RQ3 How much does the F-measure of CLM improve compared to VSM?

This experiment confirms that CLM is effective under certain conditions. CLM can recover links that VSM cannot, but CLM is not applicable to all data, especially when VSM sufficiently restores the links. However, CLM can recover transitive traceability links that VSM cannot directly determine when different words are used in each connected link.

D. Threats to Validity

This experiment used only one open source software, EasyClinic. However, CLM should be applicable to various situations. In the future, we plan to inspect the effectiveness of CLM for other data in software development. In addition, we connected links using 10% of the whole to secure a sufficient number of links. However, this number was determined sensuously. In the future, we plan to change the number of links.

V. RELATED WORD

Many methods have been proposed to automatically recover traceability links. Some methods use development log to find related links [17, 20, 21, 22]. This approach is effective if the development logs are well written. However, some software developments lack logs, limiting the applicability of such methods. In contrast, CLM does not require a specific document because it uses artifacts (e.g., requirements, designs, source code, and test cases). Hence, CLM may be applicable to software developments without development logs.

Other methods use the summons relations of the source code in the traceability links recovery methods [3, 7, 8, 13]. In these methods, source code using method A is relevant to source code with method A as only source code information is used. Although methods using the summons relations are applicable to many software developments, only the traceability links with source code are recovered. These methods cannot recover other traceability links. On the other hand, CLM does not require a specific artifact.

VI. CONCLUSION AND FUTURE WORK

We propose a method called CLM to recover transitive traceability links, and applied it to an open source software, EasyClinic. CLM requires a document from another software artifact to recover transitive traceability links. CLM effectively

recovered traceability links in cases where VSM did not. Thus, under these conditions, CLM is more effective than VSM.

In the future we intend to apply CLM to other data. Additionally, we plan to vary the number links used in order to determine the suitable number of links to connect the traceability links recovered in CLM. Moreover, this concept of connecting links may be applicable with other suggested methods. We intend to investigate the effects of applying CLM with other link recovery methods. These future studies should improve the functionality and applicability of CLM.

REFERENCES

[1] A. Czauderna, M. Gibiec, G. Leach, Y. Li, Y. Shin, E. Keenan, and J. Cleland-Huang, "Traceability challenge 2011: Using tracelab to evaluate the impact of local versus global idf on trace retrieval," In 6th TEFSE'11, volume 6, Honolulu, HI, USA, May, 2011.

[2] A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella, "Improving ir-based traceability recovery using smoothing filters," in Proceedings of the 19th International Conference on Program Comprehension, pp.21-30, June, 2011.

[3] A. Ghabi, and A. Egyed, "Code patterns for automatically validating requirements-to-code traces," the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE'12), pp.200-209, 2012.

[4] A. Kannenberg, H. Saiedian, "Why software requirements traceability remains a challenge," The Journal of Defense Software Engineering, pp.14-19, 2009.

[5] A. Panichella, C. McMillan, E. Moritz, D. Palmieri, R. Oliveto, D. Poshyvanyk, A. De Lucia. "When and How Using Structural Information to Improve IR-Based Traceability Recovery". 17th European Conference on Software Maintenance and Reengineering (CSMR), 2013.

[6] B. Dit, E. Moritz, and D. Poshyvanyk, "A tracelab-based solution for creating, conducting, and sharing feature location experiments," In 20th IEEE ICPC'12, p.203-208, Passau, Germany, June, 2012.

[7] C. McMillan, D. Poshyvanyk, M. Revelle, "Combining textual and structural analysis of software artifacts for traceability link recovery," Traceability in Emerging Forms of Software Engineering, 2009. TEFSE'09. ICSE Workshop on. IEEE, May, 2009.

[8] C. McMillan, M. Grechanik, D. Poshyvanyk, C. Fu, and Q. Xie, "Exemplar: A source code search engine for finding highly relevant applications," IEEE Transactions on Software Engineering, vol. 99, pp.1069-1087, August, 2011.

[9] E. Keenan, A. Czauderna, G. Leach, J. Cleland-Huang, Y. Shin, E. Moritz, M. Gethers, D. Poshyvanyk,J. Maletic, J. H. Hayes, A. Dekhtyar, D. Manukian, S. Hussein, and D. Hearn, "Tracelab: An experimental workbench for equipping researchers to innovate, synthesize, and comparatively evaluate traceability solutions," In 34th IEEE/ACM ICSE'12, pp 1375–1378, June, 2012.

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

[11] G. Salton, A. Wong, and C. S. Yang, "A Vector Space Model for Automatic Indexing," Communications of the ACM, vol.18, no.11, pp.613–620, November, 1975.

[12] G. Salton and M. J. McGill, "Introduction to modern information retrieval," McGraw-Hill, New York, June, 1983.

[13] H. Eyal-Salman, A.-D. Seriai, and C. Dony, "Feature-to-code traceability in a collection of software variants: combining formal concept analysis and information retrieval," the 14th IEEE International Conference on Information Reuse and Integration (IRI'13), pp.209-216, August, 2013.

[14] J. Cleland-Huang, A. Czauderna, A. Dekhtyar, G. O., J. Huffman Hayes, E. Keenan, G. Leach, J. Maletic, D. Poshyvanyk, Y. Shin, A. Zisman, G. Antoniol, B. Berenbach, A. Egyed, and P. Maeder, "Grand challenges, benchmarks, and tracelab: Developing infrastructure for the software traceability research community," In 6th TEFSE2011, May, 2011.

[15] J.H. Hayes, A. Dekhtyar, and S.K. Sundaram, "Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods", IEEE Trans. Software Eng., vol. 32, no. 1, Jan. 2006.

[16] J. Ramos, "Using tf-idf to determine word relevance in document queries." Proceedings of the first instructional conference on machine learning, December, 2003.

[17] M. Kassab, O. Ormandjieva, and M. Daneva, "A metamodel for tracing non-functional requirements," WRI World Congress on Computer Science and Information Engineering (CSIE'09), vol.7, pp.687-694, March, 2009.

[18] O. Gotel and A. Finkelstein, "An analysis of the requirements traceability problem," in Proc. of 1st International Conference on Requirements Engineering, pp.94-101, April, 1994.

[19] P. Mäder and A. Egyed, "Assessing the effect of requirements traceability for software maintenance," the 28th IEEE International Conference on Software Maintenance (ICSM'12), pp.171-180, September, 2012.

[20] R. Tsuchiya, H. Washizaki, Y. Fukazawa, K. Oshima, and R. Mibe, "Interactive recovery of requirements traceability links using user feedback and configuration management logs," Proceedings of 27th International Conference on Advanced Information Systems Engineering (CAiSE'15), June, 2015.

[21] R. Tsuchiya, H. Washizaki, Y. Fukazawa, T. Kato, M. Kawakami, and K. Yoshimura, "Recovering Traceability Links between Requirements and Source Code in the Same Series of Software Products," Proceedings of 17th International Software Product Line Conference (SPLC '13), Tokyo, August, 2013.

[22] R. Tsuchiya, H. Washizaki, Y. Fukazawa, T. Kato, M. Kawakami and K. Yoshimura, "Recovering traceability links between requirements and source code using the configuration management log," IEICE Transactions on Information and Systems, Vol.98-D, 2015.

[23] T. Dasgupta, M. Grechanik, E. Moritz, B. Dit, and D. Poshyvanyk, "Enhancing software traceability by automatically expanding corpora with relevant documentation," the 29th IEEE International Conference on Software Maintenance (ICSM'13), pp.320-329, September, 2013.

[24] CoEST, http://www.coest.org/