Trace Matrix Analyzer (TMA)

Wenbin Li, Jane Huffman Hayes, Fan Yang, Ken Imai, Jesse Yannelli, Chase Carnes, Maureen Doyle[1]
Computer Science
University of Kentucky
Lexington, Kentucky, USA
wenbin.li@uky.edu, hayes@cs.uky.edu, fan_yang_1@brown.edu, jesse.yannelli@uky.edu, chase.carnes@uky.edu,
[1] Northern Kentucky University
Lexington, Kentucky, USA
doylem3@nku.edu

***Abstract*—A Trace Matrix (TM) represents the relationship between software engineering artifacts and is foundational for many software assurance techniques such as criticality analysis. In a large project, a TM might represent the relationships between thousands of elements of dozens of artifacts (for example, between design elements and code elements, between requirements and test cases). In mission- and safety-critical systems, a third party agent may be given the job to assess a TM prepared by the developer. Due to the size and complexity of the task, automated techniques are needed. We have developed a technique for analyzing a TM, called Trace Matrix Analyzer (TMA), so that third party agents can perform their work faster and more effectively. To validate, we applied TMA to two TMs with known problems and golden answersets: MoonLander and MODIS. We also asked an experienced software engineer to manually review the TM. We found that TMA properly identified TM issues and was much faster than manual review, but also falsely identified issues for one dataset. This work addresses the Trusted Grand Challenge, research projects 3, 5, and 6.**

***Index Terms*—Formal Specification, Temporal Requirements, Translation, Requirement Comprehension, Trusted Grand Challenge, Research Projects 3, 5, and 6.**

I. Introduction

"Requirements assurance aims to increase confidence in the quality of requirements through independent audit and review" [1]. A Trace Matrix (TM) represents the relationship between software engineering artifacts and is foundational for many assurance techniques such as criticality analysis, change impact analysis, and regression testing. In a large project, a TM might represent the relationships (trace links) between thousands of elements of dozens of artifacts (for example, between design elements and code elements, between requirements and test cases). In mission- and safety-critical systems, a third party agent may need to assess a TM prepared by the developer. There are currently no automated techniques to assist such an agent.

To support assurance activities, trace links and TMs must possess a number of characteristics (shared with requirements and requirement sets, as a matter of course [2]). Trace links must be: correct, unambiguous, and verifiable; the TM must be complete, consistent, and modifiable [2]. Our work focuses on ensuring that TMs and their trace links are complete and correct. Informally, a complete trace matrix is one where all the parent level elements trace to all appropriate children elements. A correct trace matrix is one that does not contain inappropriate or spurious trace links [1]. Theoretically, it is not possible to determine if a trace matrix is complete, just as it is not possible to determine that a set of requirements are complete. We therefore move to a surrogate line of inquiry: can we develop automated techniques to evaluate completeness (and correctness) as well as human analysts.

Toward that end, we developed a tool in C++ to analyze a given trace matrix and look for six types of potentially incorrect links (listed from hardest to easiest to detect): possible "bad" links, possible missing links, parents without children (e.g., high level requirements without links), children without parents, children with too many parents, and parents with too many children (e.g., a high level requirement may have more than ten children while most of the other high level requirements have less than five). The results were promising and the tool, Trace Matrix Analyzer or TMA, was rewritten in C# and converted to a TraceLab component.

The research question addressed by this paper is: Can a technique be developed to analyze provided trace matrices at least as well as humans? The contribution of the paper is several-fold:

- Introduces a technique for analyzing a provided TM,
- Undertakes an empirical study to evaluate the technique using a publicly available dataset,
- Undertakes an anecdotal study of manual review of a TM, and
- Provides a composite TraceLab component for use by others.

We applied TMA to two trace matrices for which issues had already been identified manually by independent experts. We used two common information retrieval (IR) measures to assess the composite component (using a gold standard or answerset against which to compare): recall, a coverage measure that indicates the percentage of true trace matrix issues that were retrieved; and precision, a noise measure

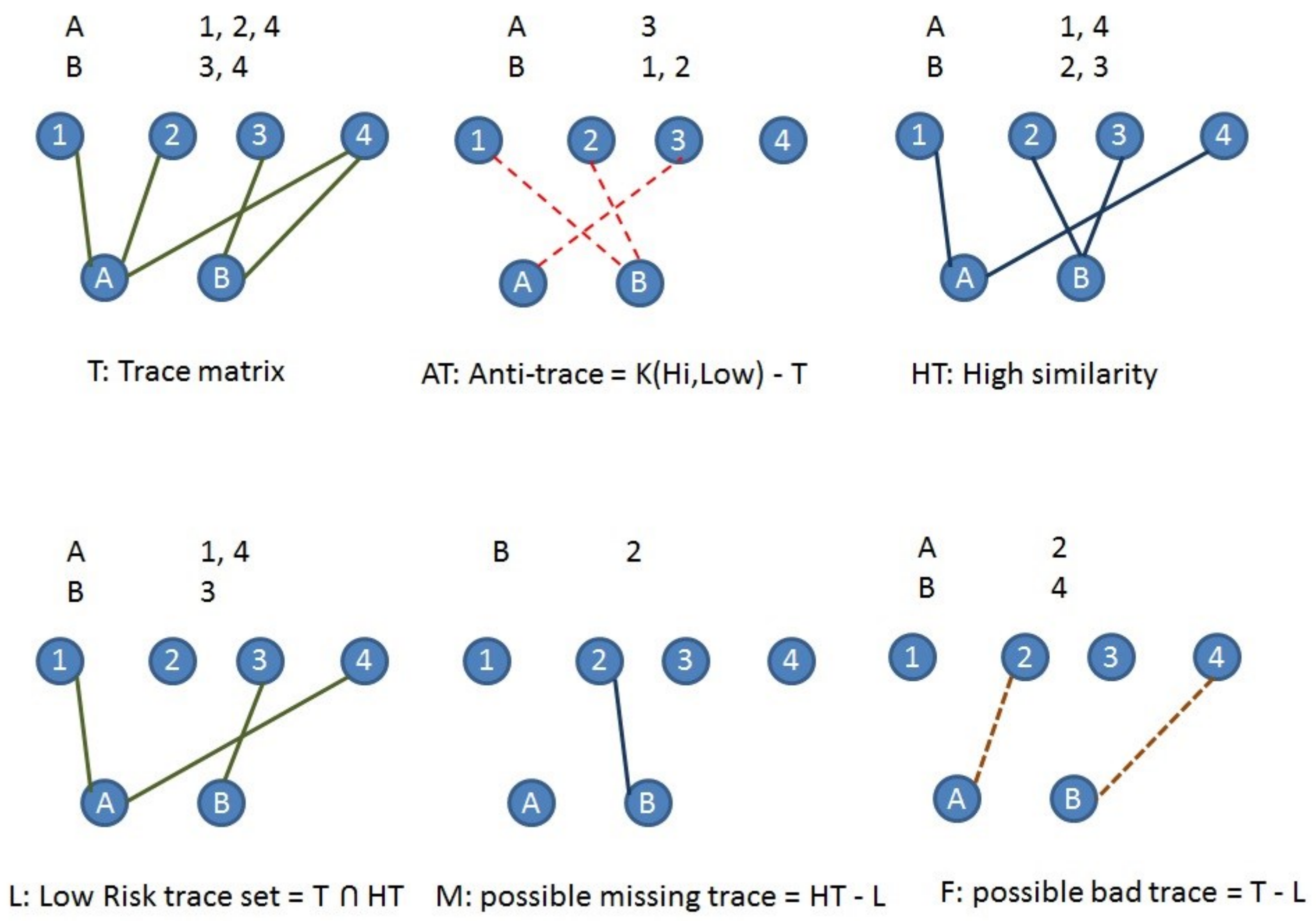

A 1, 2, 4
B 3, 4
1
2
3
4
A
B
T: Trace matrix
A 3
B 1, 2
AT: Anti-trace = K(Hi,Low) - T
A 1, 4
B 2, 3
HT: High similarity
A 1, 4
B 3
L: Low Risk trace set = T ∩ HT
B 2
M: possible missing trace = HT - L
A 2
B 4
F: possible bad trace = T - L

Figure 1. The Trace Matrix Analysis Approach (after [1]).

that indicates the percentage of retrieved issues that were correct. We also looked at execution time. TMA exhibited 95% recall and 78% precision and ran very quickly (less than 2 seconds in wall clock time) for one dataset.

We also undertook an anecdotal study to compare the tool's performance to that of a human analyst. We found little difference in the effectiveness of the two methods (TMA, manual), but found that TMA is much more efficient than manual evaluation.

The paper is organized as follows. Section 2 addresses trace matrix analysis. Section 3 presents related work. Sections 4 and 5 discuss validation and results, respectively. Section 6 provides conclusions and a look at future work.

II. TRACE MATRIX ANALYSIS

The Trace Matrix Analysis tool builds on prior work by Port et al. [1]. We explain possible issues that a TM may possess and the approach that is implemented in TMA.

A. Possible Issues

Trace Matrices must be examined by independent agents to ensure that the trace generation process was undertaken properly (whether a developer generated the TM as the lifecycle proceeded or an automated tool was used to generate the TM after the fact) as well as to check for common trace matrix issues. Specifically, our work focuses on assuring that trace matrices are complete and that individual links are correct.

In order to assure that a given TM is complete, we seek to address three questions: 1) do all parent elements have children elements?, 2) do all children elements have parents?, and 3) are there any missing links? The first question can be answered in a trivial way by examining the trace matrix for parent element identifiers (IDs) followed by no links. After inverting the trace matrix (examining it from the children element perspective), the second question can also be answered trivially by examining the trace matrix for children element identifiers (IDs) followed by no links.

The third question requires an evaluation of the trace matrix to apply heuristics or methods for identifying missing links. One way to accomplish this is to examine sibling relationships. For example, if children elements 1 and 2 share the parents A and B and child element 2 also has parent C, it could be inferred that child element 1 should also have a link to parent element C (see Figure 2). We have implemented such "sibling" checks for possible missing links. Further, we have looked at the notion of investigation sets as implemented by Port et al. to study the relationships between non-functional and functional requirements in a generated trace matrix [1].

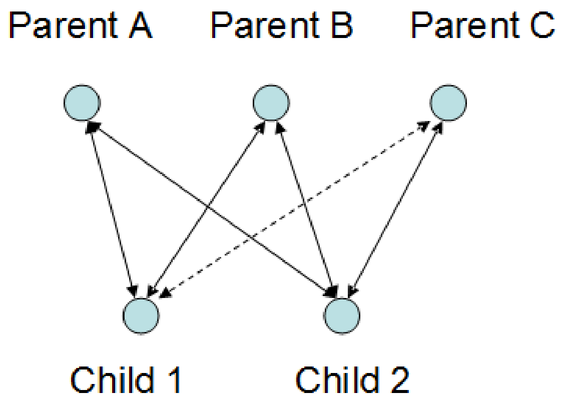


Figure 2. "Sibling" Checks

Port et al. [1] automatically generate a trace matrix between all functional and non-functional requirements in a given system. They then use heuristics to identify investigation sets for the trace matrix. This approach is shown in Figure 1. The generated trace matrix is shown in the upper left of the figure and labeled T. Its inverse is generated, called AT or anti-trace. Links that are found to possess the characteristic of "high similarity" (which Port et al. [1] do not define, but we define as a high relevance weight (from the vector space model with tf-idf weighting that is associated with a link in the TM) form the high similarity or HT set. Based on these sets, we can find the low risk investigation set or L (lower left corner of Figure 1), possible missing links or M and possible bad traces or F. Our TMA component uses these sets. M augments the set of possible missing links (completeness). F becomes the set of possible bad traces, thus addressing the other TM issue of interest: correctness of links. Next, we discuss how the approach has been implemented as a TraceLab component.

B. Analysis Approach

We implemented our approach using TraceLab, as shown in Figure 3. We created eight TraceLab components: High Similarity Matrix, Low Risk Trace, Possible Missing Link, Possible Bad Trace, Anti-Trace, Generate Issue List, Read Golden Answerset, and Compare Issue Lists. High Similarity Matrix uses the Vector Space Model to generate candidate links between the source and target artifacts. It then applies a threshold to the relevance weight and accepts all links above that threshold as elements of the High Similarity set HT. Low Risk Trace takes HT as input as well as the trace matrix T (from the answer set importer component) and determines the intersection of the two sets, L.

Possible Missing Link takes HT and L as input and returns their difference (M in the Port et al. paper [1]). Possible Bad Trace takes T and L as input and returns their difference (F, per Port et al. [1]). Anti-Trace returns AT which is the inverse of the trace matrix, T. Generate Issue List outputs all the issues identified (possible missing link, parent without children, etc.) along with the associated parent and/or child identifiers. Figure 4 shows an example issue list. Read Answerset accepts the gold standard issue list for the given trace matrix in order to evaluate TMA. Compare Issue Lists calculates recall and precision by comparing the output issue list of Generate Issue List with the golden answerset. All other components used are provided with TraceLab.

III. RELATED WORK

A number of researchers have described the creation [4, 10] or maintenance [8] of trace matrices, however validation and verification of the matrices has, up to now, primarily been performed by humans.

The use of automated methods for TM assessment was first explored and reported by A. Dekhtyar, Hayes, Sundaram, Holbrook, and O. Dekhtyar [4]. They present a method using Information Retrieval (IR), multiple trace recovery tools, and voting among the tools to demonstrate the detection and rejection of false positives introduced by automatic trace tools. They found that humans were better at finding false positives; however automation did find false positives that were not detected by humans. The findings from Dekhatyar et al. [4] influenced the requirements for a low false positive rate for the tool discussed within this paper. Also, our results have led us to consider using a variety of trace techniques for generating and combining multiple HT sets. This remains future work.

Port, Hayes, Huang, and Nikora [1] examined the problem of missing requirements between non-functional requirements and functional requirements. This paper applies a similar methodology using text-mining and statistical analysis to analyze TMs and identify sets of potential missing links for a larger set of artifacts.

Ghabi et al. [9] describe an interesting tool they developed for validation of requirements to code traces. They demonstrate their tools effectiveness using four gold-standard case studies. Ghabi's work is similar to the work described here; however, it was developed to address maintenance of the trace matrix and not overall verification and validation. In addition, differences include: TMA is a static tool, TMA does not require a caller/callee relationship, and TMA does not require a code graph for evaluation. In addition, though Ghabi's work may be extended to apply to a general graph, TMA was designed to support any and all types of artifact pair(s).

IV. VALIDATION

In order to evaluate the TMA approach, we undertook a small scale study. The research question, variables, hypotheses, study design, and threats to validity are presented below.

A. Research Question

The research question for the study is: Can a technique be developed to analyze provided trace matrices at least as well as humans in terms of effectiveness (recall, precision) and efficiency (time)?

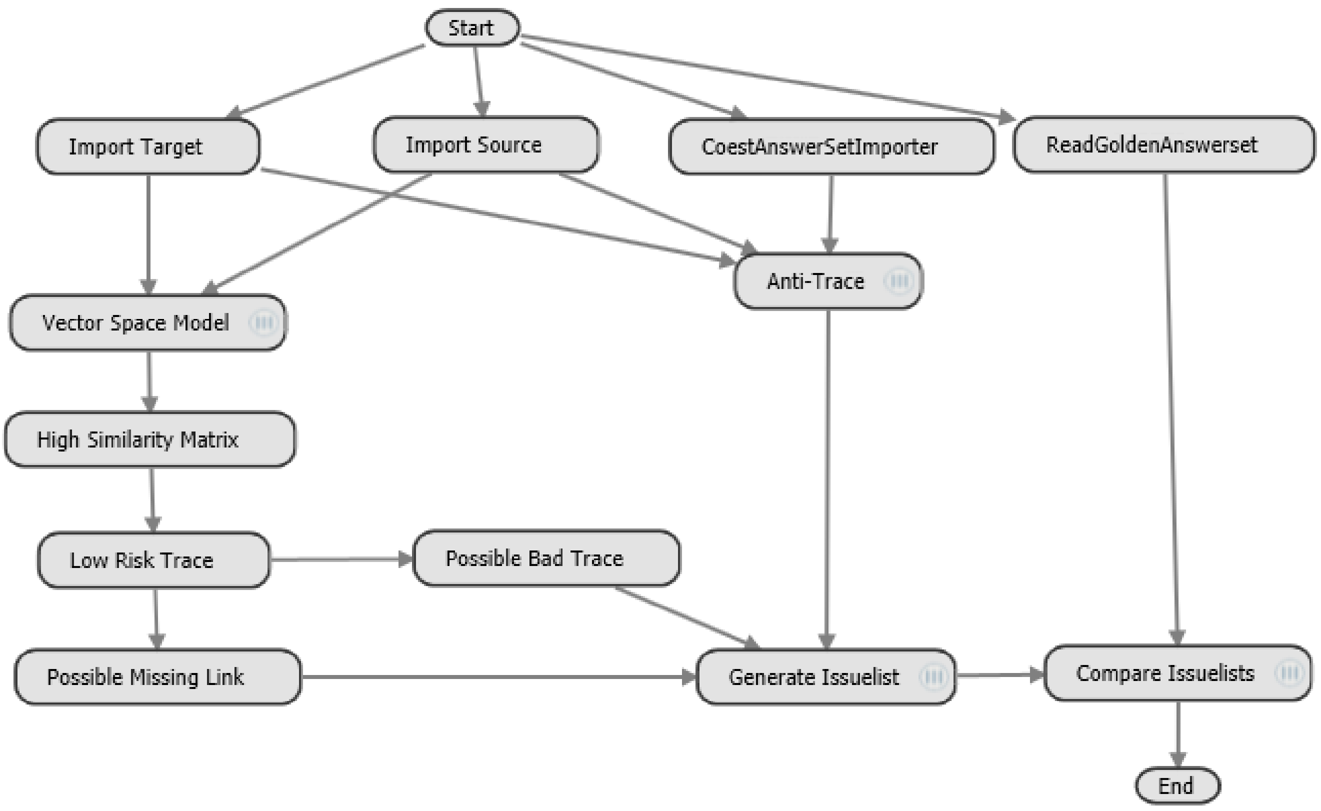
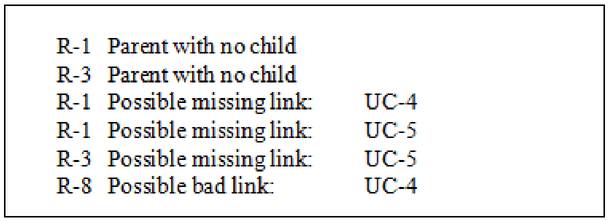


Figure 3. The Trace Matrix Analysis Composite TraceLab Component.

A. Dependent and Independent Variables

The Dependent Variables (DVs) are recall, precision, and time. Recall (R) measures the percentage of the issues that TMA is able to retrieve. Precision (P) measures how many incorrect issues TMA retrieves. Time (S) measures, in wall clock seconds, how long it takes for TMA to generate an issue list.

The Independent Variable (IV) is technique. The IV has two levels: TMA (T) or manual (M).



R-1	Parent with no child	
R-3	Parent with no child	
R-1	Possible missing link:	UC-4
R-1	Possible missing link:	UC-5
R-3	Possible missing link:	UC-5
R-8	Possible bad link:	UC-4

Figure 4. Sample Issue List

B. Hypotheses

There are three sets of hypotheses:

Null hypothesis 1 (H_{01}): R(TM) = R(M)

Alternative hypothesis 1 (H_{A1}): R(TM) > R(M)

Null hypothesis 2 (H_{02}): P(TM) = P(M)

Alternative hypothesis 2 (H_{A2}): P(TM) > P(M)

Null hypothesis 3 (H_{03}): S(TM) = S(M)

Alternative hypothesis 3 (H_{A3}): S(TM) > S(M)

B. Study Design

The study was designed to evaluate the performance of TMA on publicly available datasets compared to a human analyzing the trace matrix for the same datasets.

We ran the study on the Moderate Resolution Imaging Spectroradiometer (MODIS) dataset and the MoonLander dataset. The MODIS dataset [5, 6] is an open source NASA scientific instrument; it consists of 19 high level and 49 low-level requirements. The MoonLander dataset [7] is a text-based game written by undergraduates at CalPoly San Luis Obispo; it consists of 10 high level requirements and 5 test cases.

The consenting participant (per the University's institutional review board process) was given a pre-study questionnaire to gauge any prior experience with tracing and trace matrices. The participant possessed strong software engineering experience (17 years), which put bias in favor of the manual method. The study was then explained to the participant. The participant was given the MoonLander dataset in hardcopy. The dataset contained the trace matrix, requirements, and test cases. The participant was asked to record the time expended on the task, to record issues, and then to answer several short post-study questions (such as describing the process applied).

After finishing, the participant was given the MODIS dataset and was also shown the RETRO.NET tool [8] that could be used to interactively examine the datasets and matrix. Some hardcopy artifacts were also provided: trace matrix, high-level requirements, low-level requirements. We manually calculated recall and precision using golden answer sets for each dataset. In addition, we ran the TMA Tracelab component on the same datasets and captured execution time. The choice of threshold significantly affects the performance of TMA. We used three thresholds (0.1, 0.2, and 0.4) (filtering the similarity scores generated by the vector space model) to generate the high similarity matrix which is used to generate the list of possible bad links and possible missing links. We checked the high similarity matrix generated by the Vector Space Model component manually and found that most of the links have weights between 0.1 and 0.2. Thus, the three thresholds we used were representative.

C. Threats to Validity

There were four possible types of threats to validity for the study. A threat to internal validity was the possibility that something other than our independent variable was impacting recall, precision, and time. The main threat was possible distractions or confounding factors regarding the human analyst. We did not monitor the analyst during the study and it is possible that they were distracted by other events and/or made avail of other sources to assist them (though we do not think this was the case). In addition, we provided an automated tool to assist with review of the second trace matrix. To mitigate this internal validity threat, we asked the participant to document the process they used as well as to time their work.

A possible threat to construct validity was "hypothesis guessing." The participant may have tried to guess what the study was about and based his/her behavior accordingly. A threat to external validity is the use of two datasets that were rather small, although the MODIS dataset is a real dataset. Our datasets covered only two domains, we cannot generalize the results to all domains or all project trace matrices.

A possible threat to conclusion validity is that proper statistical analysis is not performed or that data violate the assumptions of the statistical tests. We cannot claim that our results are statistically significant (sample is too small). We discuss the results next.

V. RESULTS

Below we present the results of the study as well as some observations.

A. Study Results

The MODIS dataset and its trace matrix were inspected manually and independently: the trace matrix has 19 issues that comprise the trace matrix analysis golden answerset. Of these 19, there are: 11 parent artifacts without any children artifacts, six bad links, and two missing links. Table I shows the issues found by the participant and TMA. The participant identified 11 parent artifacts without children, four missing links, and three incorrect links. With threshold 0.1, TMA found 11 parent artifacts without children, 85 missing links, and seven bad links. TMA found all the parents with no children regardless of the threshold (this is a trivial check). With threshold 0.2, the number of missing links greatly reduced to four, while the number of bad links increased to eight. With threshold 0.4, TMA did not find any missing links and bad links increased to nine.

TABLE I. MODIS ISSUE IDENTIFICATION: TMA VS. MANUAL METHOD

	All	Parents w/out Children	Missing Links	Bad Links
Trace Matrix	19	11	2	6
Manual	18	11	4	3
TMA, 0.1	103	11	85	7
TMA, 0.2	23	11	4	8
TMA, 0.4	20	11	0	9

Table II shows the precision, recall, and time for the manual and TMA techniques applied to the MODIS dataset. The participant (the manual method) performed better than TMA with respect to finding missing links because both missing links were found; of note is that the participant also misidentified two links as missing (false positives). TMA at threshold of 0.1 also found both of the missing links, but returned many incorrect missing links. The reason for this is that there are too many links with weight higher than 0.1, and all these links are considered "*possibly missing*." For bad links, the two methods performed similarly. The participant only found half of the bad links, but all that were found were correct; TMA threshold 0.2 and TMA threshold 0.4 found all six of the bad links, but identified false positives as well. The threshold of 0.4 also prevented the approach from finding any missing links, because the weights of most links in this dataset are below 0.4.

We added the number of issues (regardless of their types) for each method and arrived at the recall and precision shown in Table III. As can be seen, the manual method has the highest precision (89% versus 85%), and TMA at threshold of 0.2 has recall as high as 95% (versus 84% for manual).

While the effectiveness of the manual method and TMA 0.2 are not so different, there is a major difference in their efficiency. The participant spent 2,280 seconds (38 minutes) to evaluate the matrix (after training was performed) while the TMA method took only one second to generate a similar result. It should be noted that any false positives generated by TMA may require review (and thus time) on the part of an analyst.

Neither the manual review nor TMA performed well on the "toy' dataset (MoonLander). All of the non-trivial issues identified by TMA (there were five in total) were false positives. In addition, TMA missed three possible missing links. There was little consolation that TMA correctly identified the two parents with no children (making for two of eight issues correctly identified (25% recall) with five false positives of seven issues identified (28.7% precision). The participant also performed poorly on this dataset, incorrectly identifying one bad link and 19 missing links for recall of 100% and precision of 20% (this took him 25 minutes or

TABLE II. PRECISION, RECALL, AND TIME FOR MANUAL AND TMA METHODS ON MODIS (ISSUES SEPARATED)

	Parents w/out Children		Missing Links		Bad Links		All Issues		Time
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	
Manual	**1.00**	**1.00**	**0.50**	**1.00**	**1.00**	0.50	**0.71**	0.63	2,280 sec
TMA, 0.1	**1.00**	**1.00**	0.02	**1.00**	0.71	0.83	0.08	**0.88**	**1 sec**
TMA, 0.2	**1.00**	**1.00**	0.25	0.50	0.75	**1.00**	0.58	**0.88**	
TMA, 0.4	**1.00**	**1.00**	-	0.00	0.67	**1.00**	0.67	0.67	

1500 seconds). We are currently investigating the dataset to understand this result. Our current thinking is that the elements of MoonLander are very simple (sentences) and there is a tremendous amount of repeated text in each element of both artifacts.

TABLE III. PRECISION, RECALL, AND TIME FOR MANUAL AND TMA METHODS ON MODIS (ISSUES COMBINED)

	Recall	Precision
Manual	84.21%	88.89%
TMA, 0.1	94.74%	17.48%
TMA, 0.2	94.74%	78.26%
TMA, 0.4	89.47%	85.00%

Based on the results, we are not able to reject null hypotheses 1 and 2. It does appear that TMA saves significant time. If we had a larger sample size, we might be able to reject null hypothesis 3.

B. Observations

It is obvious that the threshold greatly affects the performance of TMA. This is not surprising, because threshold directly affects the HT. As expected, with an increase of threshold comes a decrease in the number of links in HT. Consequently, TMA finds fewer possible missing links and more possible bad links. The recall will increase with the threshold, but the precision will reach a maximal value at a certain threshold and then decrease. In this study, the threshold value for maximal precision appears to be close to 0.2.

According to Table I, this effect is more obvious in finding missing links. While a suitable threshold (in this case, 0.2) can generate reasonable results, a higher threshold (in this case, 0.4) prevents TMA from finding any possible missing links. On the contrary, TMA 0.1 found many incorrect missing links. The low precision makes the result less useful since a human will be required to weed out false positives. To investigate this, we checked the similarity matrix generated by the Tracer component of TraceLab. We found that the weights of a large portion of the links are between 0.1 and 0.2. With a threshold of 0.1, these links were all included in the high similarity matrix and were considered as possible missing links. However, a threshold of 0.4 resulted in only one link being in the high similarity matrix, causing TMA to not find any possible missing links.

To analyze how to improve the performance of TMA, we also examined the possible missing links that were not found by TMA 0.2 and the false positive bad links that were only found by the TMA method. We found that the children artifacts of the two missing links share very few keywords with their parent artifacts; this causes the weight of these two links to be lower than the threshold of the high similarity matrix. The participant found these missing links based on the element semantics (they read the parent and child text and realized that the text meant the same thing though common terms were not used). However, the threshold prevents these links from being included in HT, which also makes it impossible to find these links in M, the possible missing links. Additionally, high thresholds also explain the two false positive bad links. The similarity weight of these two links was so low that even TMA 0.1 did not include them in HT.

The results show that the quality of HT significantly affects the performance of TMA. There are two factors that affect the quality of HT: the threshold and the tracing technique. The "best" threshold that maximizes TMA effectiveness for finding bad links and other issues may depend on the dataset. The value of this threshold should not be too low in order to prevent low precision in finding possible missing links; also, it should not be too high to prevent too many false positive bad links.

In this study, we used the vector space model (VSM) with term frequency-inverse document frequency (TF-IDF) weighing as the tracing technique. One possible way to improve HT is to use other tracing techniques (such as LSI, Probabilistic[1]) or to use Okapi or LTU as the weighting option for VSM. Using different techniques to generate multiple HT sets, it is possible to build a "combined HT" that may have higher quality.

VI. CONCLUSIONS AND FUTURE WORK

We believe that TMA can be useful in validating a given trace matrix and be used to assist in finding issues in it. With a well-designed TMA program, analysts need to merely set the proper threshold and then check the results. A suitable threshold can be estimated by checking the weights of all possible links. Alternatively, when the TMA returns too many possible missing links or possible bad links, it is clear that the threshold should be increased or decreased accordingly. Comparing the effects of various thresholds is an easy task because of the efficiency of TMA. Once the analyst gets a result, he/she should focus on the possible bad links with low weights because these links may be false positives. Similarly, he/she should focus on the possible missing links with high

[1] The reader is referred to the chapter on Information Retrieval techniques in the "Software and Systems Tracebility" book (Springer, 2012) for definitions of LSI, LTU, and Okapi

weights because these will always be included in the HT (regardless of correctness).

Future work includes examining the threshold issue to see if more specific, dataset-specific guidance can be given to human analysts. Another possible improvement is expanding the current approach by comparing the parents that have similar children. For example, if two parents share most of their children, it is possible that they also share other non-linking children. This may illustrate missing links. In addition, it should be noted that several of the analyses performed require only the TM and a list of the identifiers of the source and target artifacts (we assume that a TM presents the trace links from the source to the target [3]). This is useful for third party agents who are not permitted to share the text of the two artifacts being traced and further are not permitted to install external tools on their computer systems. At this time, we require the full text of both artifacts. Implementing checks using just the identifiers remains future work.

ACKNOWLEDGMENTS

This work is funded in part by the National Science Foundation under NSF grants CCF-0811140 (research) and ARRA-MRI-R2 500733SG067 (benchmark development). We thank our anonymous participant.

REFERENCES

[1] D. Port, A, Nikora, JH Hayes, and LiGuo Huang, "Text Mining Support for Software Requirements: Traceability Assurance ", System Sciences (HICSS), 2011 44th Hawaii Inernational Conference on, February 2011.

[2] OCZ Gotel, FT Marchese, and S. Morris, “On Requirements Visualization,” In Proceedings of the Second International Workshop on Requirements Engineering Visualization (REV’07).New Delhi, India: IEEE Computer Society, 15-19 October 2007.

[3] OCZ Gotel, JC Haung, JH Hayes, A Zisman, A Egyed, P Grunbacher, A Dekhtyar, G Antoniol, J Maletic, and Patrick Mader, Traceability Fundamentals, in Software and Systems Traceability, Jane Cleland-Huang, Orlena Gotel, Andrea Zisman, editors, Springer, London, New York, 2012, p. 6.

[4] Alex Dekhtyar, Jane Huffman Hayes, Senthil Sundaram, Ashlee Holbrook, Olga Dekhtyar, “Technique Integration for Requirements Assessment.” In Proceeding of the IEEE Int’l Conference on Requirements Engineering, 2007.

[5] [5] MODIS Science Data Processing Software Requirements Specification Version 2, SDST089, GSFC SBRS, November 10, 1997.

[6] MODIS Requirements Specification, SDST-0591, GSFC SBRS, September 11, 1997.

[7] Dalbey, John, MoonLander dataset, CalPoly San Luis Obispo.

[8] Jane Huffman Hayes, Alex Dekhtyar, Senthil Karthikeyan Sundaram, E. Ashlee Holbrook, Sravanthi Vadlamudi, and Alain April, "REquirements TRacing On target (RETRO): Improving Software Maintenance Through Traceability Recovery," Innovations in Systems and Software Engineering: A NASA Journal (ISSE), Vol. 3, No. 3, pp. 193-202, 2007.

[9] Ghabi, A., Egyed, A., “Code Patterns for Automatically Validating Requirements-To-Code Traces.” In Proceeding of the 27th IEEE/ACM International Conference on Automated Software Engineering, September 2012, Essen, Germany.

[10] Antoniol, Giuliano, et al. "Recovering traceability links between code and documentation." Software Engineering, IEEE Transactions on 28.10 (2002): 970-983.