Trustrace: Improving Automated Trace Retrieval Through Resource Trust Analysis

Nasir Ali
DGIGL, École Polytechnique de Montréal, Canada
E-mail: nasir.ali@polymtl.ca

***Abstract*—Traceability is a task to create/recover traceability links among different software artifacts. It uses resources, such as an expert, source and target document, and traceability approach, to create/recover traceability links. However, it does not provide any guidance that how much we can trust on available resources. We propose Trustrace, a trust-based traceability recovery process, to improve expert trust on a recovered link and trust over the traceability inputs. Trustrace has three sub components, in particular, Link trust improver (LTI), traceability factor controller (TFC), and a hybrid traceability approach (HTA). LTI uses various source of information, such as temporal information, design documents, source code structure, and so on, to increase experts' trust over a link. To develop TFC, we will perform a systematic literature review and empirical studies to find out which factors impact the traceability-process inputs and document these factors in a trust pattern. TFC trust pattern will help practitioner and researchers to know which steps they can take to avoid/control these factors to improve their trust on these inputs. In the HTA, we will combine different traceability recovery approaches. All approaches have different positive and negative points, we will combine all the positive points of different approaches to increase experts' trust over the HTA. In Trustrace, HTA will implement the LTI model following TFC instructions to improve the expert trust over recovered link as well as precision and recall.**

***Keywords*—Traceability, trust-based model, hybrid approach, quality factors**

I. Introduction

Existing comprehension models share the idea that program comprehension occurs in a bottom-up manner [1], a top-down manner [2], or some combination thereof. They also agree that developers use different types of knowledge during program comprehension, ranging from domain-specific knowledge to general programming knowledge. Traceability links between code sections and related documentation aid both top-down and bottom-up comprehension [3].

Traceability is the degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship with one another; for example, degree to which the requirements and design of a given software component match [4]. Traceability has received much attention over the past couple of decades. Researchers have proposed many approaches, *e.g.*, information retrieval-based [5], event-based [6], hypertext-based [7], scenario-based [8], goal-based [9], and rule-based [10], to recover traces among different software artifacts. These approaches use mainly three inputs for traceability recovery (TR), high-level documents, low-level documents, and experts' opinions. To the best of our knowledge, all proposed traceability recovery approaches (TRA) have either low recall and precision values or both. Several researchers [11], [12] have compared various TRA in terms of precision and recall.

In this research proposal, we claim that (i) *only improving traceability recovery approaches in themselves cannot help in improving precision and recall; we should also control the factors that impact the inputs of these approaches*. Some researchers [13], [14] have mentioned factors that impact TRA inputs. However, these factors and their impact on TRA inputs have not received considerable attention so far. If we assume that a TRA provides high precision and recall, but that its inputs have poor quality, then it will produce poor results. Thus, it is important to provide better inputs to achieve better results. We also claim that (ii) *only similarity value or single evidence to verify a link is not sufficient, more evidence that a link is true or false can help an expert to quickly make the correct decision to accept or reject a link*, and (iii) *combining goal-based, IR-based, and rule-based approaches can trace functional, non-functional requirements, improve precision and recall, and allow an expert to define custom rules for recovering traceability links.*

II. Typical Problematic Scenario

To understand how experts trust a traceability approach and various factors impact TRA inputs, let us consider a scenario where a project manager receives a verification and validation task. To complete her task, she needs up-to-date traceability links between requirements and source code. She uses IR-based TRA to automatically create traces between requirements and source code. To best our knowledge, IR-based approaches are not useful to trace non-functional requirements. IR-based approach use textual similarity to create links. If there is no textual similarity then IR-based will miss many links and produce false links. Therefore, an expert must manually remove false links and create missing links. The project manager assigns this task to an expert who has 10 years of Java and C++ experience.

Let us now further assume that source code is in Perl and that the source code identifiers' quality is poor and that

a non-professional person wrote/updated the requirements. Then, it is likely that the expert would miss some correct links again and allow false links. The expert has only textual similarity values to trust over a links, a high similarity value shows high-textual similarity between source code and requirement documents. Therefore, the automated generated traceability links after expert's verification would include incorrect links.

Let us analyse that why this scenario produces incorrect links even using good resources. It is quite possible a Java or C++ developer reads Perl code but he may not understand the code flow. Developer used meaningless identifiers name (*i.e.*, i, j, and k) that cause low similarity between high-level and low-level documents. If the similarity value is very low then the expert may not trust a link and discard that link or she has to manually read the Perl code. In addition, the expert has to manually trace non-functional requirements.

This scenario highlights the problem that how much project managers can trust on their available resource to create/recover traceability links. The project manager can produce better results for traceability if she has a process that can guide her by providing her with the level of trust on different resources, such as TRAs, experts, and recovered links.

To deal with this problem, we propose a trust-based traceability process that (i) *provides an expert more related information with similarity value to improve trust over a link*, (ii) *a hybrid approach to trace functional and non-functional requirements and improve precision and recall*, and (iii) *traceability factor controller that measures the impact of a factor (i.e., expert knowledge) on the traceability recovery process (TRP) and provides preventive measures to control/avoid the impact of that factor.*.

III. Related work

Modifying software systems require a detailed understanding of their functionalities. It is important to traverse the development artefacts looking for their relationship to develop this understanding. Traceability has been recognised as an important task in software development. Traceability has gained importance and its topics have become subject to research. Traceability relations can improve the quality of the systems being developed and reduce development time and cost [15].

Ramesh [16] separated traceability users into two groups, high-end and low-end. High-end users are organizations, which use traceability as well-defined system development policies. They customize their tools to provide better support for traceability. Low-end users use traceability relations to allocate requirements to system components. However, they do not see traceability as an important task in their development process. Low-end users do not capture process-related traceability information.

Maletic *et al.* [17] proposed a XML based traceability query language, TQL. TQL supports queries across multiple artefacts and multiple traceability link types. TQL has primitives to allow complex queries construction and execution support. The authors executed complicated pattern matching queries using XPath on srcML with a speed of 20KLOC/sec.

Maider [18] *et al.* re-focused attention on practical ways to apply traceability information models in practice to encourage wider adoption of traceability. The authors highlighted the typical decisions involved in creating a basic traceability information model, suggested a simple UML-based representation for its definition, and illustrated its central role in the context of a modelling tool.

Arkley *et al.* [19] proposed Traceable Development Contract (TDC). The TDC formalises the interaction of two development teams by defining their behaviour with respect to the state of their common development artefacts. The TDC framework records traceability information to reduce incomplete, inaccurate, and out-of-date traceability links.

Cleland-Huang *et al.* [20] presented a model-based approach designed to help organizations gain full benefit from the traceability links that they developed and to allow project stakeholders to plan, generate, and execute trace strategies in a graphical modeling environment.

Sherba *et al.* [21] proposed an approach, TraceM, based on technique from open hypermedia [22] and information integration. TraceM manages traceability links between requirements and architecture. An open hypermedia system enables the creation and viewing of relationship in heterogeneous systems. TraceM allows the creation, maintenance, and viewing of traceability relationships in tools that software professionals use on a daily basis.

Jirapanthong *et al.* [15] presented a rule-based approach to support the automatic generation of traceability relations between feature-based communality and variability analysis documents. The authors defined a traceability reference model with nine different types of traceability relations for eight types of documents. This approach classifies rules into two groups: direct and indirect rules. The rules are represented in an extension of XQuery. The authors also provided the XTraQue prototype tool that supports their approach. It allows the creation of new traceability rules and execution of these rules to verify their correctness. When a user is satisfied with a new rule, it can be inserted in traceability containing document.

IV. Research Goal

The goal of our research work is threefold. First, it deals with identifying the factors impacting TRP, measuring the impact of each factor, and documenting each factor with their severity level. Second, using temporal information, such as CSV/SVN logs, bug reports, mailing list, and so on, to find evidences that recovered link is trustworthy and building a trust model to calculate how much an expert can trust a link. Lastly, combining various TRA to build a hybrid

approach to improve precision, recall, and allow experts to define custom rules for recovering traceability links.

V. Research Questions

To the best of our knowledge, all existing TRAs [5], [6], [7], [8], [10] face low precision and recall. Researchers are proposing new approaches to increase precision and recall. However, our claim in this proposal is that improving *only* traceability approaches cannot improve traceability quality. There is a need to provide better quality process for software artefact traceability. Thus, the main question of this research is: *How to build a traceability recovery process that takes into account the quality factors and use external information to increase experts trust over links?*

We divide our main research question into sub questions:

1) *What are the existing traceability recovery process (TRP) available?*
2) *What kind of quality factors impact TRP?*
3) *Does available traceability recovery process provide support to control the quality factors?*
4) *How to measure the potential factors' impact on TRP?*
5) *How to increase an expert trust over a recovered link?*
6) *Which internal/external information could be used to increase expert trust over TRP?*
7) *Can a hybrid traceability recovery approach provide better results than existing TRAs?*

VI. Methodology

The following steps provide a details of the proposed methodology to answer the research questions.

A. Link Trust Improver

Link trust improver (LTI) uses various source of information, such as temporal information, design documents, source code structure, and so on, to improve the expert trust over a link. For example,to include temporal information, such as CSV/SVN, bug reports, mailing lists, and so on, we develop a link trust improver (LTI) model [23] that takes into account temporal information and calculates trust over a recovered link. We build the LTI on top of Web trust models [24]. LTI uses any traceability recovery approach to obtain a set of traceability links' ranked list, which rankings are then re-evaluated using a set of other available evidence in other source of information. In particular, we use the following steps, (i) *create links links between high-level documents and low-level documents (H2L)*, (ii) *trace high-level documents to other source of information (O2T)*, (iii) *verify each H2L link to see if it also occurs in O2T* (iv) *keep the link from H2L where O2T also confirms that link*. LTI answers the sub-research questions 5 and 6.

B. Traceability Factor Controller

We perform a detailed literature review on existing TRP to answer research question 1. Traceability factor controller (TFC) answers the sub-research questions $2, 3$, and 4. In TFC, we identify and document factors that impact traceability recovery approaches' inputs; second, we identify metrics/tools to measure/improve the quality of the inputs wrt. the identified factors; third, we provide precautions to control these factors; fourth, we empirically prove and quantify the impact of identified factors on the traceability recovery approaches' inputs. To achieve the first two objectives, we perform a systematic literature review of traceability recovery approaches and identify and document their key inputs and the factors impacting these inputs. We analyse the reported results in the literature for the identified factors to address our third objective. We conduct empirical studies to assess the impact of identified factors on TRP. We document all the identified factors and preventive measures to measure/improve the quality of TRA inputs using a consistent pattern, described in Table I.

Table I
Pattern to document TRA inputs, factors, and preventive measures

Attribute	Description
TRA Input	Brief introduction to the TRA input
Factor Name	Name of the factor impacting the TRA input
Definition	Definition of the factor
Scenario	A scenario illustrating the impact of the factor
Literature Review	Literature evidence of the impact of the factor
Preventive Measures	Metrics, tools, and precautions to measure and control the factor

To perform valid empirical research on these factors, we use precision and recall to measure each factor's impact on TRP. The severity of each factor will help project managers to make decisions for controlling/avoiding all or some factors.

C. Hybrid Traceability Approach

Every traceability approach has positive points. For example, goal-based traceability approaches help to trace non-functional requirements, while IR-based approaches provide better support for functional requirement traceability. We perform empirical studies on three recovery approaches: IR-based [5], goal-based [9], and rule-based [10] approaches, to propose hybrid approach. The hybrid approach helps to trace functional and non-functional requirements of systems as well as improving the precision and recall. The goal-based traceability approach's softgoal interdependency graph is used to capture non-functional requirements, while IR-based approach is combined to trace functional requirements. Last, the rule-based approach is combined with the previous two approaches to allow experts to define custom rules for recovering traceability links, to answer research question 7.

D. Validation

To evaluate the effectiveness of our proposed process, we perform empirical case studies and experiments. We

compare our proposed process with existing available TRP and TRAs. We use precision and recall for the comparison.

VII. Acknowledgment

I am deeply thankful to my supervisors for their continue guidance and brainstorming mentoring.

References

[1] N. Pennington, *Comprehension Strategies in Programming. In: Empirical Studies of Programmers: Second Workshop. G.M. Olsen S. Sheppard S. Soloway eds.* Englewood Cliffs, NJ: Ablex Publisher Nordwood NJ, 1987.

[2] R. Brooks, "Towards a theory of the comprehension of computer programs," *International Journal of Man-Machine Studies*, vol. 18, pp. 543–554, 1983.

[3] A. De Lucia, M. Di Penta, R. Oliveto, and F. Zurolo, "Improving comprehensibility of source code via traceability information: a controlled experiment," in *Program Comprehension, 2006. ICPC 2006. 14th IEEE International Conference on*, 2006, pp. 317 –326.

[4] Ieee, "Ieee std 610.12-1990(r2002)," *IEEE Standard Glossary of Software Engineering Terminology*, 1990.

[5] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering traceability links between code and documentation," Piscataway, NJ, USA, pp. 970–983, October 2002.

[6] J. Cleland-Huang, C. K. Chang, and M. Christensen, "Event-based traceability for managing evolutionary change," *IEEE Transaction Software Engineering*, vol. 29, no. 9, pp. 796–810, 2003.

[7] S. A. Sherba, "Towards automating traceability: an incremental and scalable approach," Ph.D. dissertation, Boulder, CO, USA, 2005.

[8] A. Egyed and P. Grünbacher, "Automating requirements traceability: Beyond the record & replay paradigm," in *ASE'02: Proceedings of the 17th IEEE international conference on Automated software engineering*. Washington, DC, USA: IEEE Computer Society, 2002, p. 163.

[9] J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya, and S. Christina, "Goal-centric traceability for managing non-functional requirements," in *ICSE '05: Proceedings of the 27th international conference on Software engineering*. New York, NY, USA: ACM, 2005, pp. 362–371.

[10] G. Spanoudakis, A. Zisman, E. Prez-Miana, and P. Krause, "Rule-based generation of requirements traceability relations," *Journal of Systems and Software*, vol. 72, no. 2, pp. 105 – 127, 2004.

[11] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. D. Lucia, "On the equivalence of information retrieval methods for automated traceability link recovery," in *ICPC*, 2010, pp. 68–71.

[12] A. Abadi, M. Nisenson, and Y. Simionovici, "A traceability technique for specifications," *International Conference on Program Comprehension*, pp. 103–112, 2008.

[13] G. Antoniol, "Recovery of traceability links in software artifacts and systems," Ph.D. dissertation, Montreal, Canada, 2003.

[14] J. H. Hayes and A. Dekhtyar, "Humans in the traceability loop: can't live with 'em, can't live without 'em," in *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, ser. TEFSE '05. New York, NY, USA: ACM, 2005, pp. 20–23. [Online]. Available: http://doi.acm.org/10.1145/1107656.1107661

[15] W. Jirapanthong and A. Zisman, "Xtraque: traceability for product line systems," *Software and Systems Modeling*, vol. 8, no. 1, pp. 117–144, February 2009.

[16] B. Ramesh, "Factors influencing requirements traceability practice," *Commun. ACM*, vol. 41, no. 12, pp. 37–44, 1998.

[17] J. I. Maletic and M. L. Collard, "Tql: A query language to support traceability," in *TEFSE '09: Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 16–20.

[18] P. Mader, O. Gotel, and I. Philippow, "Getting back to basics: Promoting the use of a traceability information model in practice," in *TEFSE '09: Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 21–25.

[19] P. Arkley and S. Riddle, "Overcoming the traceability benefit problem," in *RE '05: Proceedings of the 13th IEEE International Conference on Requirements Engineering*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 385–389.

[20] J. Cleland-Huang, J. H. Hayes, and J. M. Domel, "Model-based traceability," in *TEFSE '09: Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 6–10.

[21] S. A. Sherba, K. M. A., and M. Faisal, "A framework for mapping traceability relationships," in *2nd International Workshop on Traceability in Emerging Forms of Software Engineering at 18th IEEE International Conference on Automated Software Engineering*, Montreal, Quebec, Canada, 2003, pp. 32–39.

[22] K. Osterbye and U. K. Wiil, "The flag taxonomy of open hypermedia systems," in *HYPERTEXT '96: Proceedings of the the seventh ACM conference on Hypertext*. New York, NY, USA: ACM, 1996, pp. 129–139.

[23] N. Ali, Y.-G. Guéhéneuc, and G. Antoniol, "Trust-based requirements traceability," in *ICPC '11: Proceedings of the International Conference on Program Comprehension (ICPC'11)*. Washington, DC, USA: IEEE Computer Society, 2011.

[24] D. H. McKnight, V. C., and C. K., "The impact of initial consumer trust on intentions to transact with a web site: a trust building model," *The Journal of Strategic Information Systems*, vol. 11, no. 3-4, pp. 297 – 323, 2002.