Humans in the Traceability Loop: Can't Live With 'Em, Can't Live Without 'Em

Jane Huffman Hayes and Alex Dekhtyar

ABSTRACT

The human analyst is required as an active participant in the traceability process. Work to date has focused on automated methods that generate traceability information. There is a need for study of what the analysts do with traceability information as well as a study of how they make decisions.

1. INTRODUCTION

In examining the traceability problem in software engineering, one fact becomes apparent: the human is in the loop. When developing embedded hyperlinks in related software engineering artifacts, the developer can always override the tool's linking decisions. In assessing the accuracy of a given requirements traceability matrix (RTM), an analyst can disagree with existing links. In building an RTM for two textual artifacts, an analyst can identify links not found by a linking tool (completeness analysis).

In the end, the analyst can always override any prior decision and has the FINAL say on whether or not the traceability is correct. In the case of RTMs that are built for safety or mission critical software systems and/or are part of a software safety case (such as those required for certification of an instrumentation and control software system for a nuclear power plant or an air traffic control software system), the human analyst MUST make the final decision on whether or not high level elements in an RTM have been fully satisfied by low level elements linked to it.

Current and prior research in traceability (with the exception of [5]) ignores this fact. Instead, the research focuses on the role of the computer in traceability. That is to say, given two textual artifacts, how accurately can a given method or technique trace them to each other? In all fairness, one should not study the impact of the human analyst until it is shown that automated methods by themselves are sufficiently accurate. But the results reported recently in [4, 6, 10, 1, 7, 2, 3] suggest that using methods from the arsenal of information retrieval and data mining for the automated tracing is both accurate and efficient. Despite that, we know that automated methods and techniques produce imperfect results. It is now time to study **WHAT happens when a human analyst becomes involved and makes decisions concerning the computer.s traceability output**. We believe this is a fundamental issue in traceability research and this paper addresses: (a) our position on the issue; (b) evidence supporting our position; and (c) suggestions for future research in the area.

The rest of the paper is organized as follows. In Section 2, we express our position concerning the need to study analyst interaction with tools in tracing tasks. In Section 3, we discuss some preliminary results we have obtained in our prior studies and elaborate on their significance. In Section 4 and 5, we discuss the questions that the proposed research will address in the future.

2. POSITION

As mentioned in [5], we forsee two different ways in which tracing methods can be used to study/improve software engineering practices. The first way involves the after-the-fact study of the software development process artifacts with the purpose of learning and improving the process for the future projects. The second way involves introduction of advanced, automated tracing techniques into the software development process for the purpose of improving **current** process. These two ways present somewhat different sets of challenges for software engineering researchers interested in traceability. Traditional traceability research tends to address the concerns of the first approach more than those of the second.

Our position in this paper is that the unique challenges posed by the second approach, which assumes introduction of automated tracing methods directly into the software development process (for example at the Validation & Verification, or Independent Validation & Verification stages), **must be analyzed and addressed** by the emerging research on traceability. Two that extent, we make two important observations.

First, we maintain that anything with traceability for emerging forms of software engineering MUST have the human in the loop. Automated methods rooted in information retrieval, data mining, machine learning, and/or natural language processing produce approximations, estimates of the final traceability. It is the job of a human analyst, especially when tracing is part of a mission-critical software project, to confirm and possibly correct computer-generated traces.

Building on this observation, the **second** part of our position is that the traceability community must study human interaction (reaction) with the results produced by automated methods. Do human analysts improve predictions made by automated methods? What factors affect the work of human analysts with the automated method results? Do human analysts trust results produced by automated methods? And last, but not least, *what can we do to improve the **final** traces* (the ones produced by human analysts after examining the candidate traces produced automatically)?

3. EVIDENCE

How do we know that this problem exists? In its most rudimentary forms, traceability requires retrieval of relevant information. Only a human can make the final determination of whether or not retrieved items are indeed relevant. As mentioned in Section 1, there are a myriad of examples in traceability where analysts make decisions about provided links. *The key issue is that analyst decisions are not always correct*. Anecdotal evidence obtained for [4] (see also citemsr05) showed that a human analyst could commit both errors of omission (throwing correct links out of the trace) and errors of commission (adding incorrect links to the trace) while examining the trace. As a result, the final accuracy of the trace was not a significant improvement over the computer-generated one (in the specific experiment, described in [4, 5], we witnessed improved precision and decreased recall). In addition, we have evidence that correct analyst feedback improves dynamic link generation [6, 10].

For [5], we obtained a few more data points concerning analyst interaction with automatically generated candidate traces. In all cases reported in [5] (and we have no other data as of this moment), we witnessed the same thing as in the case reported in [4]: a significant percentage of individual analyst decisions concerning candidate links were erroneous, regardless of the initial accuracy (measured as precision and recall) of the candidate trace.

Additionally, our own personal experience with tracing as well as our experience with professional analysts on a small tracing study [5] indicates that analysts “dislike” tracing (almost half the analysts for the tracing study dropped out). This is understandable as it is a tedious, boring, human-power intensive, error-prone task. Tools are not much assistance either. Most tools require the analyst to assign keywords to all elements of both textual artifacts being traced, and/or to build a detailed hierarchical list of keywords, and/or perform interactive string searches for potential matches or links, and/or examine lengthy, yet poor quality, lists of potential links.

As mentioned above, we have some evidence that analysts can make decisions that make the returned results worse! For example, given a list of potential matches, they throw away good links together with bad. They also keep some bad links [4, 5]. In our pilot study [5], three analysts were given three different traces of the MODIS dataset [8, 9] and asked to verify them. The traces were “doctored” to have different precision and recall as shown in Table 1 in the columns marked “Original”.

The precision and recall of the trace provided by the analysts are shown in the columns marked “Final”. As seen from this table, in all three cases, recall went down. In one case, precision was drastically reduced. Figure 1 depicts the change in accuracy graphically (dashed arrow represents the human analyst case from [4]).

So, analysts are a **vital** part of tracing, but they do not enjoy it, and they make the returned results worse! What can be done?

4. WHAT TO STUDY AND HOW TO STUDY IT?

As detailed above, our experience suggests that we need to study analyst interaction with tracing tools and determine what affects analyst performance and how. How do we get from such and open-ended goal to a specific research agenda? In this section we outline a number of hypotheses we hope to evaluate in our future work.

Clearly, we need to establish the factors that affect analyst performance. We hypothesize now that we can break these factors into three broad categories.

Tool output. Presumably, the better the accuracy of the tool, the more accurate the result of analyst inspection should be. That is to say, if the tool retrieves potential links that are not relevant, and the analyst accepts the tool.s recommendations, the resulting RTM will not be accurate. This statement, however, comes with a number of caveats. First, precision and recall are not symmetric measures from the analyst.s point of view. To improve precision, the analyst needs to spot errors of commission and remove existing candidate links from the trace. To improve recall, the analyst needs to spot errors of omission, i.e., notice that a link is missing from the trace. Practice shows that errors of omission are much harder to spot and fix. Therefore, when assessing the accuracy of the results, we assume that the analyst would favor candidate link lists with the highest possible recall and .decent. precision over candidate link lists with high precision and .decent. recall. Second, certain high-accuracy regions in the recall-precision space may actually yield decreased accuracy. In particular, under certain circumstances, we speculate that candidate link lists with 95numbers . if the analyst assumes that the results need significant improvement (there is also very little room for error here). What we are truly interested in are the .comfort regions. of the recall-precision space . the values of recall and precision that tend to make analysts significantly improve the results produced by the tool.

Subjective characteristics of the tool. The key reason why an analyst would decrease the accuracy of an excellent computer-generated candidate trace is the analyst.s lack of trust in the software that produced it. The tool may be capable of producing high-quality estimates, but as long as the analyst is intent to second-guess its results, the process will not work as desired. Conversely, if the tool produces poor estimates, but the analyst shows more trust in the tool than it deserves, the accuracy of the final trace will remain low.

It is thus important to understand the mechanisms by which the analyst decides to trust/mistrust the tool. From the examples above, it is clear that the ideal situation is for the analyst to have a clear and correct impression of the tool.s accuracy. What affects the analyst perception of the tool? We speculate that in addition to the objective accuracy of the tool, subjective reasons play an important role here as well. In particular, the issues of analyst interaction with the tool and the convenience of the user interface have to be studied. At the outset we see two key sets of factors affecting the analyst trust: (1) the quality of the graphical user interface and the convenience of the tool.s use, and (2) the transparency of the tool.s linking decisions.

We are currently working on version 2 of REquirements TRacing On-target (RETRO), a special-purpose requirements tracing tool. Our plan is to test the interface design solutions for ease-of-use and convenience and determine what features the tracing tool GUI must have. Among the features eventually implemented in RETRO will be the ability for the analyst to examine the reasons for including a link into a candidate link list. The tool will display matching keywords and/or other reasons for matching high- and low-level elements for each link. The tool will also allow the analyst to control how the keywords are used during feedback processing stages.

X-factors. This broad category includes factors related to the analyst him/herself. Clearly, such factors as analyst experience, level of boredom with tracing tasks, style of work (in the broadest sense), and specific timeframe in which the work was assigned can have effect on analyst interaction with the tool. This, of course, is not restricted

to tracing tasks: such factors affect a broad range of software engineering tasks. We need to determine which of these factors need to be considered, which can be factored out, and what effect the remaining ones have on the analyst performance.

5. PLAN FOR THE FUTURE

We posit that a number of steps can be taken to address this situation. First, we must study what humans do. Specifically, we must collect data on the decisions that analysts make, e.g., what percentage of the time do they throw away good links? What percentage of the time do they keep bad links? What percentage of the time do they claim that an element has been satisfied by its relevant items in the other artifact, but in fact there are missing items? Second, we need to examine the factors that may impact analyst decision-making, such as:

- Quality of the retrieved potential links; Analyst experience in tracing;
- Analyst experience with the domain;
- Analyst experience with the project;
- Analyst.s confidence in the tracing tool;
- Etc.

Taking a sample item from above, analyst experience in tracing, let us examine how future work will proceed. A controlled experiment will be designed and undertaken. For example, analysts with similar backgrounds and general experience levels but with varying degrees of experience in tracing will be separated into three groups: those with much tracing experience, those with moderate tracing experience, and those with little or no tracing experience. The analysts will be given tracing tasks to perform manually and also using automated tools. We will collect data such as: mean time to trace a high level element; final recall of trace; final precision of trace; precision of trace, etc. These measures will then be correlated to analyst experience to identify possible trends and to uncover findings of interest.

Scenarios for which we currently have data and may use for the above studies include, but are not limited to:

- Trace high level textual requirements to lower level textual requirements (and vice versa),
- Trace high level textual requirements to textual design elements (and vice versa), and
- Trace high level textual requirements to textual defect reports (and vice versa).

Acknowledgements

Our work is funded by NASA under grant NAG5-11732. We thank Stephanie Ferguson and Ken McGill. We thank the Metrics Data Program, Mike Chapman, and the MODIS program.

6. REFERENCES

[1] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering Traceability Links between Code and Documentation. *IEEE Transactions on Software Engineering*, Volume 28, No. 10, October 2002, 970-983.

[2] Jane Cleland-Huang, Raffaella Settimi, Oussama Ben Khadra, Eugenia Berezhanskaya, Selvia Christina: Goal-centric traceability for managing non-functional requirements. in *Proc. ICSE* 2005: 362-371.

[3] Alex Dekhtyar, Jane Huffman Hayes, and Tim Menzies. Text is Software Too, in *Proceedings of the Mining of Software Repositories Workshop*, Edinburgh, Scotland, May 2004, pp. 22-26.

[4] J. Huffman Hayes; A. Dekhtyar, and J. Osborne, Improving Requirements Tracing via Information Retrieval, in *Proceedings of the International Conference on Requirements Engineering (RE)*, Monterey, California, September 2003.

[5] Jane Huffman Hayes, Alex Dekhtyar, Senthil Sundaram, Text Mining for Software Engineering: How Analyst Feedback Impacts Final Results, in *Proceedings of Workshop on Mining of Software Repositories (MSR)*, associated with ICSE 2005, St. Louis, MO, May 2005.

[6] Jane Huffman Hayes, Alexander Dekhtyar, Senthil Sundaram, Sarah Howard, Helping Analysts Trace Requirements: An Objective Look, in *Proceedings ofIEEE Requirements Engineering Conference (RE)* 2004, Kyoto, Japan, September 2004, pp. 249-261.

[7] A. Marcus, and J. Maletic. Recovering Documentation-to-Source Code Traceability Links using Latent Semantic Indexing, in *Proceedings of the Twenty-Fifth International Conference on Software Engineering* 2003, Portland, Oregon, 3 . 10 May 2003, pp. 125. 135.

[8] MODIS Science Data Processing Software Requirements Specification Version 2, SDST-089, GSFC SBRS, November 10, 1997.

[9] http://promise.site.uottawa.ca/SERepository/

[10] Senthil Sundaram, Jane Huffman Hayes, Alex Dekhtyar, Baselines in Requirements Tracing, in *Proceedings of Workshop on Predictive Models of Software Engineering (PROMISE)*, associated with ICSE 2005, St. Louis, MO, May 2005.

	Original		Final	
Case	Precision	Recall	Precision	Recall
1	39.6%	60.9%	45.1%	56.1%
2	20%	90.2%	58.7%	65.8%
3	80%	29.2%	22.9%	26.8%

Table 1: Three datasets used in the experiment: starting and final precision and recall.

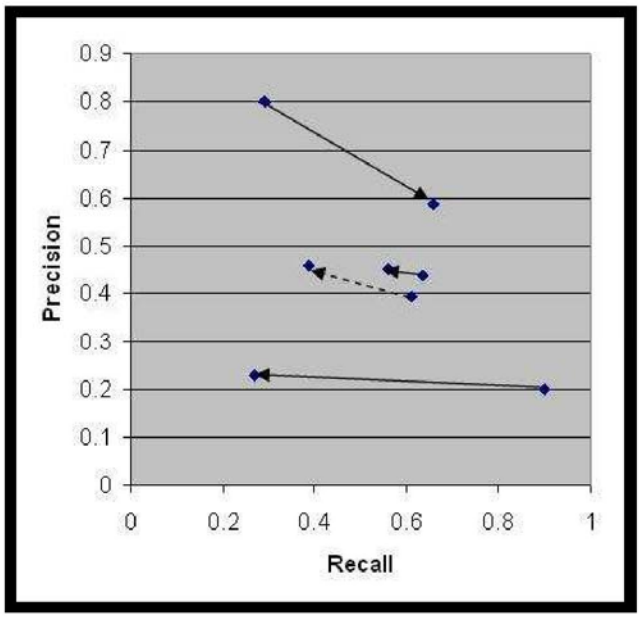

Figure 1: Analysts did not exhibit the pattern of improving presented traces.