Towards Overcoming Human Analyst Fallibility in the Requirements Tracing Process (NIER Track)

David Cuddeback
Cal Poly
dcuddeba@calpoly.edu

Alex Dekhtyar
Cal Poly
dekhtyar@calpoly.edu

Jane Huffman Hayes
University of Kentucky
hayes@cs.uky.edu

Jeff Holden
Cal Poly
jholden@calpoly.edu

Wei-Keat Kong
University of Kentucky
wkkong1@uky.edu

ABSTRACT

Our research group recently discovered that human analysts, when asked to validate candidate traceability matrices, produce predictably imperfect results, in some cases less accurate than the starting candidate matrices. This discovery radically changes our understanding of how to design a fast, accurate and certifiable tracing process that can be implemented as part of software assurance activities. We present our vision for the new approach to achieving this goal. Further, we posit that human fallibility may impact other software engineering activities involving decision support tools.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications—*tools*; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*; K.m [**MISCELLANEOUS**]: Software psychology

General Terms

Traceability, Human Factors, Software Assurance

Keywords

Traceability, Tracing, Tools, Accuracy, Validation

1. INTRODUCTION

Change impact analysis, satisfaction assessment, regression testing, reverse engineering, re-testing are all important activities in software engineering that share a commonality: a traceability matrix (TM) is required in order to effectively proceed. Unfortunately, practitioners often do not build traceability matrices, or do not keep them up to date. Tracing activities, as conducted in industry, are labor-intensive, tedious and prone to human error. In the past 10 years, the traceability research community has worked to address this problem by developing automated techniques aimed at generating traceability matrices between pairs of textual software engineering artifacts (such as between requirements and test cases, or requirements and design, etc.) [2, 12, 6, 3]. These methods, based on information retrieval, natural language processing and text mining techniques, generate candidate traces (so called until an analyst vets/approves the trace) much faster than manual tracing.

To date, the quality of a TM produced by automated methods has been mostly equated to its accuracy as measured by recall, precision and f-measure. When TMs produced by automated methods are used in follow-on activities, automated methods that produce more accurate TMs are preferable to those that produce lower accuracy TMs.

In critical applications with safety and/or environmental consequences or large amounts of revenue at stake, traceability matrices are often mandated or are a standard part of the software development and software assurance processes (verification and validation (V&V), independent V&V, safety case analysis, etc.). There, tracing must be *fast*, *accurate* and *certifiable*. The former cannot be achieved without integrating automated methods, while the latter requires that a human analyst vets the TM. In such situations, which we call in this paper *semi-automated tracing*, the overall quality of the tracing is determined by the accuracy of the TM produced by the human analyst, not the automated tool.

In 2005, we began to examine the role of the analyst in the tracing process [8, 7]. Specifically, we asked whether *higher-accuracy candidate TMs reliably lead to higher-accuracy final TMs* in semi-automated tracing scenarios. Our initial thought was that the semi-automated tracing behaved essentially in a "garbage-in - garbage-out" way: low accuracy candidate TMs would be more daunting for the analysts to deal with and would yield poor accuracy in the final TM, while highly accurate candidate TMs would "reveal" the truth to the analyst resulting in high quality final TMs. Our other informal expectation was that analysts would tend to generally improve the accuracy of the TMs during the trace validation process. We conducted a small number of observations of industry analysts performing tracing. The results suggested that our expectations might not be accurate [8].

It was not until the completion of a larger study in late 2009 [4] that we had a chance to put our hypotheses [8, 7] to the test. The results reported in the larger, multi-institution study [4] have provided more credence to our emerging doubts that the accuracy of the incoming candidate TM and the accuracy of the TM that the analyst sub-

ICSE '11 May 21-28 2011, Waikiki, Honolulu, HI, USA

mits are nearly always positively correlated. Where traceability research over the years concentrated on achieving more accurate TMs produced by automated methods in anticipation that it would lead to immediate improvement in the tracing processes, our work [4] and the follow-up studies described in this paper *paint a drastically different picture*. Based on these results we observe that *in semi-automated tracing scenarios, the analysts tend to degrade high accuracy candidate TMs much more often than they improve them.*

This observation changes drastically how we view tracing in software assurance. To ensure high accuracy of the final TM produced by the analyst, it appears that *we can no longer rely on the high accuracy of the automated method.* Rather, we must admit that while analyst involvement in the tracing process is necessary, analyst fallibility is inherent in the process. It is our intent to determine the means of overcoming it.

In this paper, we overview the experiments we conducted to study how human analysts validate traceability matrices. We discuss the key observations that we see emerging from these studies and their impact on the efforts of the traceability research community to design and implement effective tracing processes for software assurance. We present an emerging research agenda and discuss the broader implications of our discoveries.

2. WHAT WE STUDIED

In order to examine how analysts perform when validating candidate TMs, we conducted a series of studies over the course of the last 12 months. Two of the studies involved the use of tracing tools for traceability matrix delivery; in the third study, the artifacts were presented to participants in hardcopy and traced manually.

All three studies utilized the same dataset, a BlueJ plugin Java code formatter named ChangeStyle. This dataset was chosen because: (a) the domain is easily understood by study participants, and (b) its size makes trace validation tasks achievable in about one hour of time. The dataset contains 32 requirements and 17 system tests. The research team generated and validated the *golden standard* TM which contains 23 links from requirements to system tests [4].

The studies were conducted at two sites: Cal Poly and University of Kentucky. Participants in all studies were students enrolled in various software engineering courses. Each study assigned each student analyst a candidate TM and asked them to validate it. In two studies, we used the UI of two versions of the tracing tool RETRO [9, 5] to deliver the candidate TMs and facilitate validation. In a third study, all materials were provided to the participants in hardcopy and the participants validated candidate RTMs manually. The participants in all studies were asked to submit their final TM, whose accuracy was assessed against the golden standard TM. In all studies, we collected some additional information about participant prior experience, confidence, opinion of the tracing process and effort spent on it.

Over the three studies, a total of 88 participants submitted their version of the validated TM. The key results of our studies are summarized in Figures 1(a)–(f). Graphs are rendered in *recall-precision* space. Points on each graph represent the recall and precision of TMs featured in the study. Figure 1(a) shows the accuracy of candidate TMs assigned to study participants; Figure 1(d) shows the accuracy of the submitted TMs. TMs from the three studies are shown using different symbols. As seen from Figure 1(a), the accuracy of initial TMs spanned the recall-precision space[1].

Figures 1(b)–(c) and 1(e)–1(f) show each participant's performance. For clarity, we split the display by the quadrant in the recall-precision space in which the participant's initial candidate TM was located. Each participant's performance is represented by a vector with the tail indicating the accuracy of the initial (assigned) TM and the head (arrow) of the vector indicating the accuracy of the submitted TM.

3. WHAT WE LEARNED

Submitted RTMs cluster to a "hotspot". Figure 1(a) shows that we assigned candidate TMs to participants relatively evenly over the recall-precision space. At the same time, Figure 1(d) shows that the accuracy of final TMs tended to converge in a hotspot region. The hotspot was initially observed in [4] and confirmed in the follow-up studies. We define it as containing TMs with 17 (# of system tests) to 32 (# of requirements) links with accuracy represented by the f_2-measure[2] in the $[0.6, 0.75]$ interval. TM size boundaries were selected to represent the "one link per test case" and "one link per requirement" intuition, while the f_2 boundaries are "nice" approximations of values that maximize the tightness of the cluster within the boundaries. We note, however, that: *(a) not a single participant in our studies recovered the golden standard TM(!), and (b) the hotspot region falls relatively far from perfect accuracy.* The participants are achieving *some form of consensus*, but the achieved consensus *is still relatively inaccurate.*

Use of tools vs. manual tracing. Overall, similar trends (see below) were observed across all studies. Some slight differences are noted here. In the hardcopy study, there were more final TMs that fell just short of the necessary recall to be in the hotspot. However, participants using the manual tracing method more reliably produced higher precision TMs. 31 out of 38 (81.6%) participants who manually vetted the TMs produced final TMs with at least 50% precision, while 29 out of 45 (64.4%) participants using a software tool accomplished the same result.

We hypothesize that use of a tracing tool UI for trace validation helps analysts correct *errors of omission*. RETRO includes a mechanism to allow analysts to search for keywords in the artifacts, which can simplify missing link discovery. Analysts who vet the TM manually have no such support for finding missing links.

All quadrants are different. We observed different analyst behavior based on the accuracy of their starting candidate TM in [4]. Follow-up studies add additional credence to these observations. Furthermore, the studies show that the general behavior of analysts is consistent, whether they vet a TM manually or use a software tool. In the *low recall, low precision* region (Figure 1(e)), analysts exhibit the desired behavior—replacing bad links with good ones. TMs in the *high recall, low precision* region (Figure 1(f)) start with many links, so analysts focus on weeding out the bad links. This leads to higher precision. Analysts end up removing

[1]We tried to achieve even distribution of assigned TMs within the recall-precision space. Any observed unevenness is due to non-responses.

[2]Harmonic mean of recall and precision, favoring recall, as we consider capturing all true links in a TM as being more important than not admitting false positives.

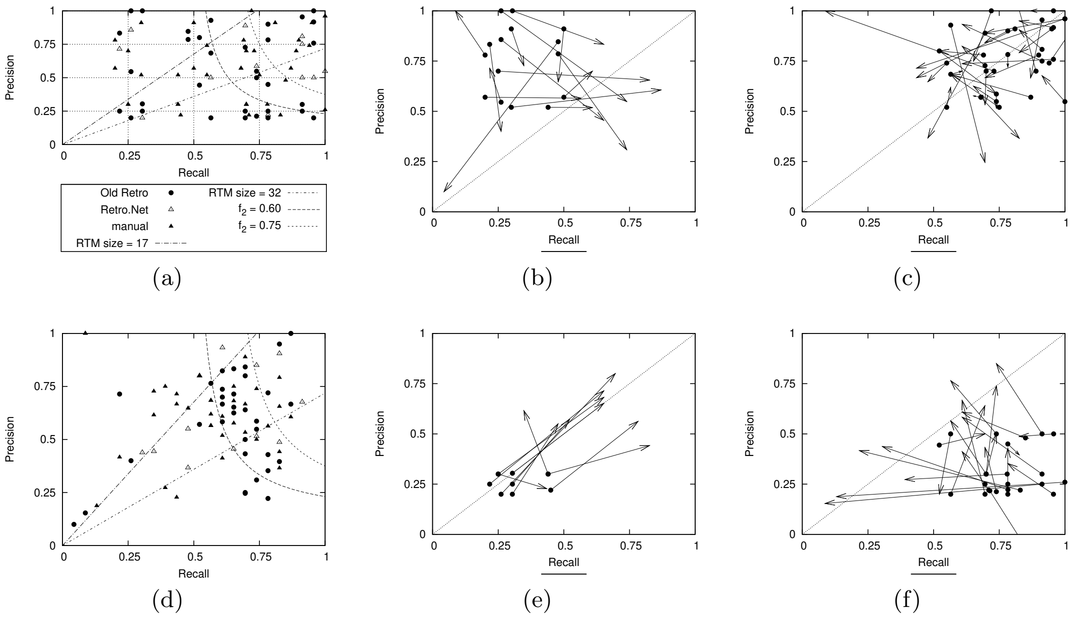

Figure 1: Results of our traceability study. (a), (d): distribution of assigned and submitted TMs. (b),(c),(e),(f): performance of individual participants by accuracy of assigned TM.

some good links in the process, leading to stagnant or somewhat decreased recall. The opposite behavior is observed in the *low recall, high precision* region (Figure 1(b)), where TMs have very few links. Analysts tend to focus on finding more links and, in doing so, add both good (improved recall) and bad (decreased precision) links. Finally, analysts vetting TMs from the *high recall, high precision* region (Figure 1(c)) reduce the overall accuracy of the RTM, some by adding bad links and others by removing good links. In many cases, the resulting TMs are still among the most accurate ones observed, *but we want the analysts facing near-perfect candidate TMs to **improve**, not degrade the accuracy of the TM*, and we are *not observing* such behavior.

We are in the process of conducting a thorough statistical analysis of the results to find the best predictors for the *accuracy of the final TM* and the *change in accuracy.* However, even with the limited data shown in the Figure 1 graphs, we can see that the accuracy of the initial TM (or the cluster it is in) is a strong predictor for both factors. As participants failed to recover the golden standard TM, *we observe that human analyst fallibility in trace validation tasks **is both unavoidable and predictable**.*

4. WHAT WE CAN DO

The last observation has a profound impact on how we approach traceability research. How are we to proceed from here, with a realization that much of our prior work has perhaps focused on the wrong problem, or at a minimum has failed to properly take into account a very important variable? We address this question below, starting with a look at possible threats to the validity of our empirical work.

The *threats to validity* for our work revolve around our use of a single dataset, and on our use of students as study participants. Our initial small study [7] was conducted with software assurance analysts and a different dataset, and its results are certainly in line with the observations made here. Thus, there is some anecdotal evidence that our observations hold across projects. Most students undertaking the project were juniors or seniors, not so less experienced than human analysts performing tracing in industry: a task often relegated to the junior personnel. Note also that Host et al. found that students are suitable replacements for industry professionals if performing small tasks of judgment [10]; this is true of our work. That being said, it is clear that we must repeat our studies on different datasets, using different tracing tools, etc. We plan to repeat our study at DePaul University in early 2011. We hope to also undertake a similar study using industry professionals, perhaps in concert with a planned industry survey for the Center of Excellence for Software Traceability [11]. Our goal in each of these studies is to confirm, refute, or modify our current conclusions.

In addressing the "what now?" question, our dilemma is not so different from any "irritation" which results in the *"can't live with 'em, can't live without 'em."* thought. There are basically four possible courses of action that could be pursued. We address each below.

Eliminate or ignore the analyst. In this course of action, the research community accepts the challenge of building techniques that are not fallible. The trusted software community (safety- and mission-critical software system developers and assurers) will have to be convinced to modify their process to no longer require a human to vet the TM. The only way to achieve this is to develop tools that recover perfect traces. This approach is unlikely to succeed because: a) this has been the research community goal all along and it has yet to be achieved; b) the trusted software community may always want a human to have the last say; and c) even if the communities overcome a) and b), the question of liability for failures due to mistakes made in the traceability analysis step may never be resolved. The research agenda for this course of action shares many goals of our community articulated in the Grand Challenges of Traceability [1].

Quarantine the analyst. In this approach, traceability researchers will in essence build a "wrapper" (using a software testing analogy) around the "untrusted" component (i.e., the analyst). It may look like a firewall or a security exception that queries "Are you sure you want to trust Bob? Permanently, or just this time?" At this point we can only speculate on what we would be protecting against. The *expected* accuracy of the automatically generated candidate TM may guide specific "quarantine" instructions. For example, most automated tracing techniques produce high recall, low precision candidate TMs [2, 12, 6, 3]. For such TMs, we might make it much harder for the analyst to reject a candidate link than to accept it or add a new link. Rejection attempts would force the analyst to jump through a number of extra hoops forcing them to seriously evaluate causes for their decision. The research agenda for this approach essentially involves translating the overarching *"trust but verify"* principle into the language of tracing activities. The analyst would be viewed as an almost toxic element in the process, and the "quarantine" measures would be akin to blacklisting or other security processes that seek to prevent miscreants from attacking one's computer.

Change the analyst. Today's analysts may be fallible, but it does not mean that the analysts of tomorrow should be. One of the key Grand Challenges of Traceability [1] is Training. We ask ourselves and our community: what can we do as educators and practitioners to improve analyst preparedness and to eliminate the analyst fallibility?

Embrace the analyst. Changing the analysts may work in the long term, but in the short term, other approaches may be needed. The **embrace the analyst** course of action throws up its hands and accepts that *fallible* humans are part of the process. It sets about to systematically and in a detailed fashion unravel the mystery of the human interacting with the tracing tools/tracing process. This ambitious undertaking needs to be approached as if there is no existing knowledge about how humans work with the process. We may need to experimentally study ONE variable at a time: size of the TM, experience of the analyst; domain of the artifacts; tracing tool; accuracy of the candidate TM; etc. Only then can we have a body of evidence that will permit us to design techniques to take advantage of the strengths of humans and to avoid their weaknesses. One could imagine, for instance, a smart tracing tool that has learned its "owner's" weaknesses and can help mitigate them, much as a spell checker or grammar checker assists the writer.

Our experiments yielded a significant amount of data yet to be analyzed. We are starting to analyze and mine this data to obtain more insight into the actual process of tracing and human decision-making in it, in line with the **embrace the analyst** approach. Regardless of the need to do more studies and perform more in depth analysis of the data, it cannot be understated that this discovery may **radically, revolutionarily change how our community delivers a "good" software-supported tracing tool/process to industrial software assurance professionals**.

As a final thought, we posit that tracing is not the only area where human analysts are fallible. Further, upon discovering that analysts are predictably fallible, it seems almost irresponsible to not study the analyst and the accuracy of the final product/final decision/final result for any area/tool/technique/process that relies on the human to make the final decision or have the final say. In addition to challenging our traceability research community to elevate the study of the analyst to paramount importance, we suggest that the software engineering research community as a whole reconsider the emphasis on evaluating techniques/tools in isolation and urge them to consider the human in the loop and resulting impacts on final accuracy.

5. ACKNOWLEDGMENTS

Work at Cal Poly is funded in part by a grant from Lockheed Martin. Work at UK is funded in part by the NSF grant CCF-0811140. We are grateful to John Dalbey for providing the source for the ChangeStyle dataset and to Gene Fisher, Clark Turner and David Janzen for allowing us to conduct studies in their courses.

6. REFERENCES

[1] Grand challenges in traceability. Technical Report COET-GCT-06-01-0.9, Center of Excellence of Traceability, September 2006.

[2] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering Traceability Links Between Code and Documentation. *IEEE Transactions on Software Engineering*, 28(10):970–983, Oct 2002.

[3] J. Cleland-Huang, C. K. Chang, and M. Christensen. Event-Based Traceability for Managing Evolutionary Change. *IEEE Transactions on Software Engineering*, 29(9):796–810, 2003.

[4] D. Cuddeback, A. Dekhtyar, and J. Hayes. Automated Requirements Traceability: The Study of Human Analysts. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 231–240, Sydney, Australia, Oct 2010. IEEE.

[5] A. Dekhtyar, J. H. Hayes, and J. Larsen. Make the most of your time: How should the analyst work with automated traceability tools? In *PROMISE '07: Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, page 4, Washington, DC, USA, 2007. IEEE Computer Society.

[6] J. Hayes, A. Dekhtyar, and S. Sundaram. Advancing Candidate Link Generation for Requirements Tracing: the Study of Methods. *IEEE Transactions on Software Engineering*, 32(1):4–19, Jan 2006.

[7] J. H. Hayes and A. Dekhtyar. Humans in the Traceability Loop: Can't Live With 'Em, Can't Live Without 'Em. In *TEFSE '05: Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering*, pages 20–23, New York, NY, USA, 2005. ACM.

[8] J. H. Hayes, A. Dekhtyar, and S. Sundaram. Text Mining for Software Engineering: How Analyst Feedback Impacts Final Results. In *MSR '05: Proceedings of the 2005 International Workshop on Mining Software Repositories*, pages 1–5, New York, NY, USA, 2005. ACM.

[9] J. H. Hayes, A. Dekhtyar, S. Sundaram, A. Holbrook, S. Vadlamudi, and A. April. REquirements TRacing On target (RETRO): Improving Software Maintenance through Traceability Recovery. *Innovations in Systems and Software Engineering: A NASA Journal*, 3(3):193–202, Sep 2007.

[10] M. Host, B. Regnell, and C. Wohlin. Using students as subjects - a comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, 5(3):201–214, 2000.

[11] http://www.traceabilitycenter.org.

[12] A. Marcus and J. Maletic. Recovering Documentation-to-Source-Code Traceability Links Using Latent Semantic Indexing. In *Proceedings of the 25th International Conference on Software Engineering, 2003*, pages 125–135. IEEE, May 2003.