Automated Requirements Traceability: the Study of Human Analysts

David Cuddeback
Computer Science Department
California Polytechnic State University
San Luis Obispo, CA, United States
dcuddeba@calpoly.edu

Alex Dekhtyar
Computer Science Department
California Polytechnic State University
San Luis Obispo, CA, United States
dekhtyar@calpoly.edu

Jane Huffman Hayes
Computer Science Department
University of Kentucky
Lexington, KY, United States
hayes@cs.uky.edu

***Abstract*—The requirements traceability matrix (RTM) supports many software engineering and software verification and validation (V&V) activities such as change impact analysis, reverse engineering, reuse, and regression testing. The generation of RTMs is tedious and error-prone, though, thus RTMs are often not generated or maintained. Automated techniques have been developed to generate candidate RTMs with some success. When using RTMs to support the V&V of mission- or safety-critical systems, however, a human analyst must vet the candidate RTMs. The focus thus becomes the quality of the final RTM. This paper investigates how human analysts perform when vetting candidate RTMs. Specifically, a study was undertaken at two universities and had 26 participants analyze RTMs of varying accuracy for a Java code formatter program. The study found that humans tend to move their candidate RTM toward the line that represents $recall = precision$. Participants who examined RTMs with low recall and low precision drastically improved both.**

***Keywords*-traceability; requirements; information retrieval; decision support**

I. Introduction

Research has shown that information retrieval techniques can be effectively applied to generate candidate RTMs in an automated fashion for textual artifacts [1], [2], [3]. These methods retrieve a high percentage of related items (for example, when tracing from a specific user story to a collection of test cases, the methods find almost all of the related test cases), but also retrieve many unrelated items (false positives). This shortcoming of the automated methods has led to a plethora of research on how to decrease the number of false positives retrieved [3], [4], [5]. Research that focuses on improving automated traceability methods is often called "the study of methods" [3]. As research has progressed and practitioners have begun to use the tools developed by academia, a new area of interest has emerged: the study of the analyst.

Automated methods generate RTMs that must be vetted by human analysts. The role of the human is particularly important when the RTMs are generated to support verification and validation (V&V) and independent verification and validation (IV&V) activities for mission- or safety-critical software systems. The human analyst must vet the candidate RTM and add and remove links as necessary to arrive at the final RTM. The quality of the final RTM is of paramount concern. If automated methods generate candidate RTMs in such a way that human analysts make bad decisions and generate low quality final RTMs, the reduction of human effort is immaterial—the process will have failed. This is true even if the automated methods output perfect (or near perfect) candidate RTMs—as long as human analysts do not recognize it during the vetting process. This suggests that automated methods for generating candidate RTMs are *valuable* in such settings *only if the human analysts make the right decisions* with the information provided to them.

Our research addresses the question of *whether the analysts will make the right decisions.* To that end, this paper concentrates on the study of the analyst (rather than the study of the methods) by examining the human role in the tracing process. We posit the following research questions: (1) how do human analysts transform the requirements traceability information produced by automated methods? (2) how does the accuracy change in that process? (3) does the amount of time an analyst spends impact the quality of the results?

To examine these questions, we designed a version of our requirements tracing tool, REquirements TRacing On target (RETRO), that allowed us to present candidate RTMs of known accuracy to analysts. We then had the analysts vet the RTMs and we measured the accuracy of the final RTM. Specifically, we worked with 26 computer science and software engineering students at two different universities in the United States[1] who examined RTMs for a Java code formatter program (tracing its requirements to test cases). We report our discoveries in this paper.

The research methodology described in the paper is applicable to a wider range of tasks involving human analyst interactions with decision support software[2]. There are three key aspects of the tracing process that affected the nature of our study: (a) the presence of automated methods that provide suggestions for a specific task, (b) the need for a human analyst to examine the suggestions, and (c) the notion of accuracy associated with the produced result. Researchers can use similar approaches to study other settings within the

[1]Our work was approved by the IRB at each University.
[2]Understood in a broad sense here.

1090-705X/10 $26.00 © 2010 IEEE
DOI 10.1109/RE.2010.35

IEEE computer society

broad area of requirements engineering which exhibit these characteristics.

The rest of the paper is organized as follows: Section II covers background information on automated traceability and related research. Section III describes our research method. Section IV reports the results of our study. Section V covers the threats to our study's validity, and finally Section VI contains our concluding remarks and future work.

II. Background and Related Work

Gotel defines *requirements traceability* as "the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)" [6].

Requirements tracing is the process of establishing traceability. In general, tracing involves linking elements from a high-level artifact to elements of a low-level artifact. An *artifact* can be any by-product of a software life cycle, including a requirements document, design document, and code. An *element* is a distinct piece of an artifact that can be traced. Examples include a requirement or use case from a requirements document; a class, method, or package in source code or design documents; and a section or subsection of documentation.

The output of the tracing process is a *requirements traceability matrix* (RTM). It defines the mapping between elements of one artifact and elements of the other artifact. Any RTM that exists before the tracing process is complete is said to be a *candidate RTM*, because it is a candidate to become the final RTM. The *final RTM* is the one approved by a human analyst.

A pair of elements that trace to each other is called a *link*. A *candidate link* is any possible link between two artifacts. Thus, for two artifacts with 10 and 5 elements, there are $10 \times 5 = 50$ candidate links for the elements of the two artifacts. In order to measure the accuracy of an RTM, some notion of correctness is needed for the links in the RTM. We refer to a link that is correct as a *true link* and a link that is incorrect as a *false link*.

When textual artifacts are traced to each other, requirements tracing can be viewed as an information retrieval task. Information retrieval (IR) is concerned with which documents from a collection of documents are relevant to a query. In requirements tracing, the high-level requirements act as queries and the low-level elements represent the collection of documents [7].

A. Measures

Consider a tracing process consisting of a set of high-level requirements $\mathcal{H}$ of size M and a set of design elements $\mathcal{D}$ of size N. For a particular requirement $q \in \mathcal{H}$, let n_q be the number of candidate links between q and the design elements in $\mathcal{D}$ that an automated tool returns. Let r_q be the number of those links which are correct and R_q be the actual number of correct links between q and the elements in $\mathcal{D}$ [3].

Recall is the percentage of correct links that are found [3]. Given a requirement q, the recall for the individual requirement is $\frac{r_q}{R_q}$. The overall recall for the entire document is defined formally in (1).

$$recall = \frac{\sum_{q \in \mathcal{H}} r_q}{\sum_{q \in \mathcal{H}} R_q} \tag{1}$$

Precision is the percentage of retrieved candidate links that are correct [3]. Given a requirement q, the precision for the individual requirement is $\frac{r_q}{n_q}$. The overall precision for the entire document is defined formally in (2).

$$precision = \frac{\sum_{q \in \mathcal{H}} r_q}{\sum_{q \in \mathcal{H}} n_q} \tag{2}$$

F-measure is the harmonic mean of recall and precision. Defined formally in (3), it represents a balance between recall and precision and can be weighted to emphasize one metric or the other. $b = 1$ weights recall and precision equally, $b < 1$ favors precision, and $b > 1$ favors recall.

$$f_b = \frac{1 + b^2}{\frac{b^2}{recall} + \frac{1}{precision}} \tag{3}$$

In this paper, following Hayes, Dekhtyar, and Sundaram [3], we use the f_2-*measure* (i.e., $b = 2$), because we observe that it is easier to remove incorrect links than to find missing links and f_2 favors recall over precision.

B. Automated Traceability as Information Retrieval

Research has shown that information retrieval techniques are efficient and effective at generating candidate links [1], [2], [3]. Table I summarizes some of the current research in methods for automated candidate link generation.

Antoniol, Canfora, Casazza, DeLucia, and Merlo [1] applied term frequency-inverse document frequency (TF-IDF) and a probabilistic IR method to trace source code to documentation and requirements in two case studies. They traced C++ classes to manual pages for LEDA (Library of Efficient Data types and Algorithms). Their second dataset was a hotel management system called Albergate. They reported high levels of recall (86–100%) but low precision (6–19%) for both methods.

Marcus and Maletic [2] achieved similar results on the same datasets as Antoniol, Canfora, Casazza, DeLucia, and Merlo [1] using latent semantic indexing (LSI) to automate tracing in the opposite direction. For the LEDA dataset, LSI

Technique	Dataset	Recall	Precision
TF-IDF [1]	LEDA, Albergate	86–100%	6–18%
Probabilistic Method [1]	LEDA, Albergate	94–100%	6–19%
Latent Semantic Indexing [2]	LEDA, Albergate	91–100%	16–25%
TF-IDF [8]	MODIS [9], [10]	63%	39%
TF-IDF with Thesaurus [8]	MODIS [9], [10]	85%	40%
TF-IDF with feedback [3]	MODIS [9], [10], CM-1 [11]	90%	80%

Table I
RESEARCH IN IR TECHNIQUES FOR AUTOMATED TRACEABILITY.

scored similar recall (96–97%) but higher precision (18–25%) than TF-IDF and the probabilistic model. For the Albergate dataset, recall and precision for LSI was similar to that of Antoniol, Canfora, Casazza, DeLucia, and Merlo [1] (91–100% and 16–17%, respectively).

Hayes, Dekhtyar, and Sundaram [3] studied requirements to requirements tracing using TF-IDF, TF-IDF with a simple thesaurus, and LSI. They also studied the use of Standard Rochio feedback analysis to incorporate analyst feedback into the automated methods. The results of the study showed that applying user feedback to filter results automatically fixed some errors in the original results.

C. The Human Side of Automated Traceability

Our research direction, introduced in [12], [13], is to study the ways in which human analysts affect the final traceability results when using automated tracing tools. To our knowledge, there has been only one attempt to study the analyst's role in editing or vetting RTMs [12]. While the study involved only four analysts, it showed that analyst behavior is a problem worthy of further research [12]. The results of the study [12] are shown in Fig. 1. There, each vector represents the change affected by one analyst to an RTM. The starting point of a vector represents the recall and precision of the initial RTM given to an analyst. The end point of a vector represents the recall and precision of the RTM submitted by the analyst after performing vetting. The key observation from the study [12] is that analysts make both errors of omission (throwing out correct links) and commission (adding incorrect links) [13].

On the other side of the interface between the human and the automated tool, Dekhtyar, Hayes, and Larsen [14] simulated human analyst decision making to study different strategies that humans may utilize when working with automated tracing tools. The simulations assumed that analysts always make correct decisions about whether a candidate link is a true link or a false link. The results showed that if analysts can correctly classify candidate links, incorporating analyst feedback provides a 7–13% savings in effort for the analyst.

To date, no large-scale study of automated traceability involving human analysts has been conducted. Any evidence of human effects on automated traceability data so far

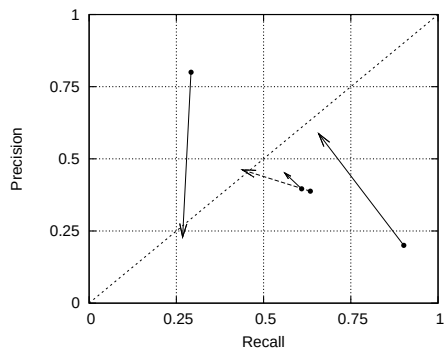


Figure 1. Results from a pilot study [12]. Arrows indicate change in accuracy after analyst corrections.

is anecdotal. Our work extends the aforementioned study [12] by conducting a more rigorous study involving more participants.

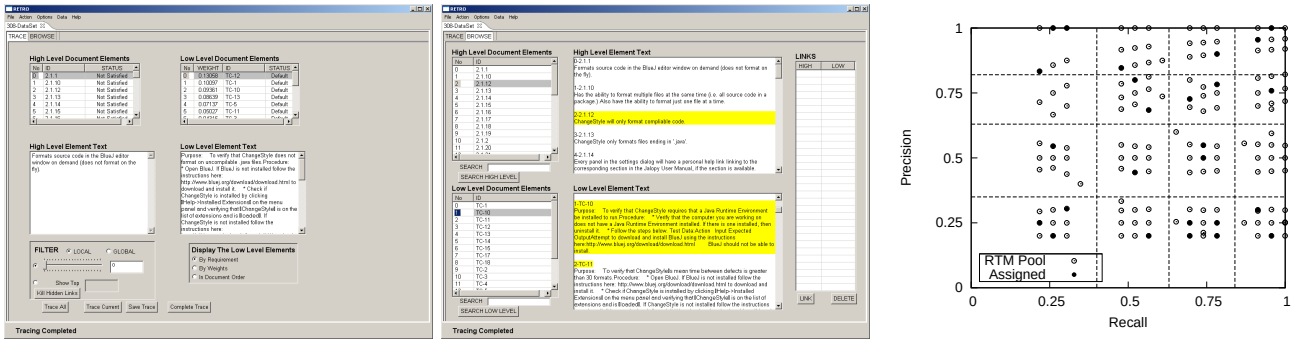
III. METHODOLOGY

For our study, we concentrated on observing what analysts do when vetting candidate RTMs obtained from automated tools. Our goal is to determine if we can better understand the work of human analysts. The key goals of our study were outlined as the research questions in Section I.

A. Research Tool

In our study, we used a modified version of RETRO (REquirements TRacing On-target) [15]. RETRO uses TF-IDF vector space retrieval to suggest candidate RTMs and uses the analyst's corrections to provide feedback to the retrieval algorithm using Standard Rochio feedback [3]. RETRO evolved from a research toolkit for tracing targeting IV&V analysts and system maintainers, and it has already been used to establish traceability between artifacts such as requirements, design documents [7], [3], and bug reports [16].

RETRO has two modes. The automated tracing mode, shown in Fig. 2 (a), lets the analyst work with the results of and provide feedback to the automated tracing methods used in RETRO. The analyst can confirm or reject any candidate link suggested by the automated methods. Rejecting a link



(a) (b) (c)

Figure 2. (a) RETRO's "trace" tab; (b) RETRO's "browse" tab; (c) the pool of candidate RTMs that can be assigned to participants.

corrects for errors of commission. The manual tracing mode, shown in Fig. 2 (b), allows the analyst to correct errors of omission.

Modifications: To study the impact that initial candidate RTM accuracy has on the accuracy of the final RTM, it is necessary to provide *different participants* with *different candidate RTMs* for the same dataset. We made two modifications to RETRO. We added a user login dialog to the startup screen. We also replaced the IR methods used to deliver candidate RTMs with a mechanism that allowed us to assign each user a specific candidate RTM.

The new candidate RTM delivery system associates specific recall and precision targets with each user ID. It starts with the candidate RTM computed by RETRO, and then adds or subtracts candidate links to reach the appropriate recall and precision values. When removing candidate links, links are removed in order of the smallest relevance score to the highest so that the strongest matches stay in the candidate RTM. When adding candidate links, a link is chosen at random and given a random, low relevance score.

B. Dataset

Our study involved two datasets. The first is the *training dataset*, which is only used for a training exercise to familiarize the participants with RETRO. The training dataset is very small, consisting of 10 functional requirements and 5 system tests, so that participants can trace it very quickly.

For the actual experiment, we constructed a dataset using a project assignment from a junior-level software engineering course sequence that spanned two quarters. Throughout the course sequence, the students produced a requirements document, written system test procedures, and an RTM for the system tests. This project was selected because: (a) its domain (a Java code formatter plugin for BlueJ) is easily understood by upper-division students in computer science or software engineering, (b) all of the documents were produced by the development team at appropriate times during the project's development, and (c) its size lends itself well to a tracing task that is reasonable to ask participants to complete in about one hour.

To limit the scope of the tracing task, we extracted only the functional and non-functional requirements from the requirements document and only the system tests covering those requirements. The result was a dataset consisting of 32 requirements (18 functional and 14 non-functional) and 17 system tests. We stripped traceability information from the system tests, and then converted all data into a format readable by RETRO. This dataset is referred to as the *experimental dataset*.

The RTM for the experimental dataset was manually verified by the research team. Since the original RTM was created by the development team *at the time that the system tests were written*, we defaulted to their decision in any cases where there was uncertainty about the traceability between two elements. The resulting RTM contained 23 links between the requirements and system tests. The verified RTM is the *golden standard RTM*, against which the accuracy of all other RTMs is measured.

C. Candidate RTM Preparation

In our study, each participant was asked to review a candidate RTM, further referred to as the *initial RTM*, with precision and recall selected from a predefined pool of possibilities. In Fig. 2 (c), we show the pool of recall-precision possibilities chosen for the study. We group the possible initial candidate RTMs into regions of similar recall and precision, which helps in assigning initial candidate RTMs to participants and analyzing the results for outcomes that depend on the accuracy of the initial candidate RTM.

The pool of candidate RTMs was generated by calculating nine points that surround the midpoint of each region. The midpoints were selected to be the inner product of $\{25\%, 50\%, 75\%, 95\%\}$ recall and $\{25\%, 50\%, 75\%, 95\%\}$ precision. The nine points in each region were calculated by taking all combinations of adding -5%, 0%, and $+5\%$ to recall and precision. We could then calculate the number

of true positives and false positives needed for each RTM by solving equations (1) and (2). Since the number of links has to be an integer, not every point calculated by this algorithm was unique. Any duplicates were discarded. In addition to the points calculated by the algorithm, points used in previous experiments were included.

D. Procedure

The study took place in four upper-division software engineering classes at two universities, and consisted of two assignments and two surveys. Participation in the study was voluntary. Students were offered 1% extra credit in their class for participating and given the option to complete an alternative extra credit assignment of equal difficulty.

We started by giving all the students a pre-experiment survey, which was designed to gauge their prior experience and comfort with tracing. After collecting the surveys, a researcher discussed requirements tracing and RETRO in a one-hour presentation/practice session. The presentation specifically covered how to work with RETRO to correct errors of omission and errors of commission to ensure that study participants had the knowledge necessary to make improvements to the candidate RTMs.

To familiarize the participants with RETRO, we asked them to trace our training dataset using RETRO. Participants were given a printout of instructions on how to use RETRO and a link to a page where they could download RETRO and the training dataset. The training exercise started during the in-class presentation, and participants were asked to complete it outside of the class. They were not required to turn in anything from the training exercise.

The research team analyzed the results from the pre-experiment surveys to determine the participants' prior experience with requirements tracing. This information was used in assignment of initial RTMs to participants. All participants were separated into two groups based on their experience and then assigned RTMs. For each group, participants were assigned to a region by round-robin, and then a random RTM within the region was assigned to the participant. Because our sample size is relatively small, a random assignment of all participants would have risked grouping experienced participants in the same region. Our assignment procedure avoided this.

One week after the practice session, students were given the experimental tracing task. Each user received a unique user ID for RETRO, used by RETRO to present the initial candidate RTM assigned to the specific user. Participants were given about one week to trace the experimental dataset outside of class time. They received written instructions for the tracing task, a time log sheet, and a link to the experimental dataset. Participants were asked to keep a record of the time they spent on the task and any issues that they encountered during the task. Submission instructions asked participants to: (a) save their final RTM and email it to the research team, (b) submit the time log to their course instructor, and (c) take an on-line post-experiment survey that asked them for their reactions to the tracing assignment (whether they felt prepared, how difficult the task was, whether RETRO benefited or hindered them in their task, etc.) and how long they spent on the assignment and the training exercise.

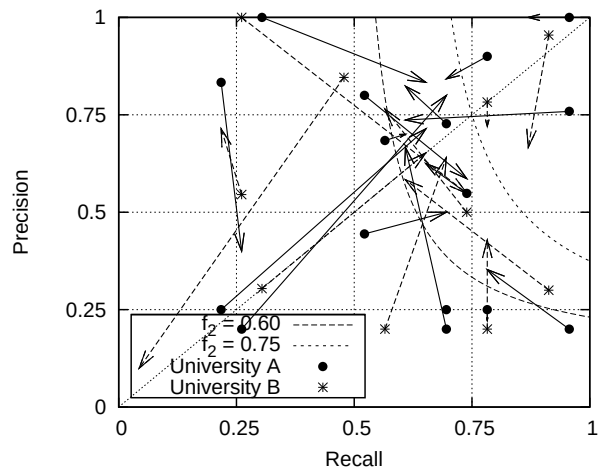


Figure 3. Change in recall and precision for all participants.

E. Data Collection

We assembled a rich set of meta-information from the pre- and post-experiment surveys and time logs in addition to the final RTMs turned in by the participants. In this paper, our independent variables are the recall, precision, and f_2-measure of each participant's initial candidate RTM, and our dependent variables are the recall, precision, and f_2-measure of each participant's final RTM and an estimate of each participant's effort spent on the task as self-reported on their time logs and post-experiment surveys. Further analysis of the data is left for future work.

IV. Results

This section presents results and analysis.

A. Overview of Results

As shown in Fig. 2 (c), we collected 26 responses to our study from four groups of participants: three software engineering courses at California Polytechnic State University (Cal Poly) and a senior project course at the University of Kentucky. One of the universities provided 10 responses and the other provided 16. The filled dots represent assigned candidate RTMs while the hollow dots represent candidate RTMs that were not assigned. Fig. 3–8 depict the key results of our study.

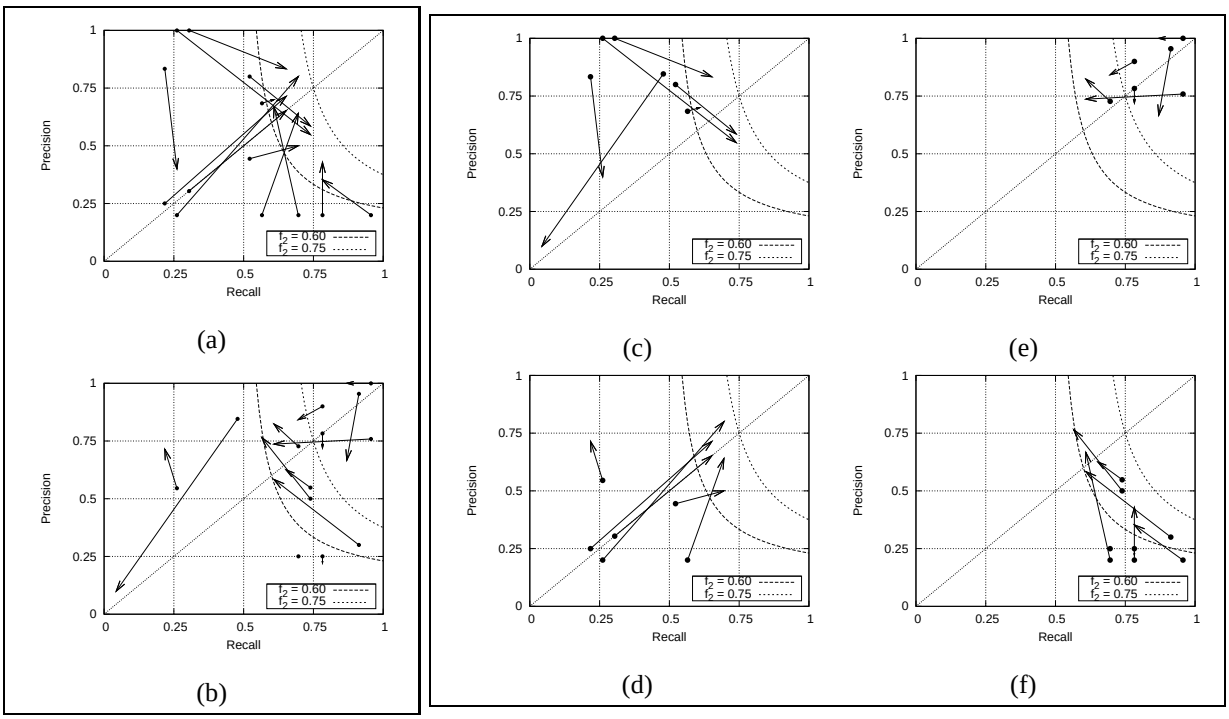


Figure 4. Change in recall and precision of participants who (a) *improved* and (b) decreased the f_2-measure, and (c)–(f) change in recall and precision by region/quadrant.

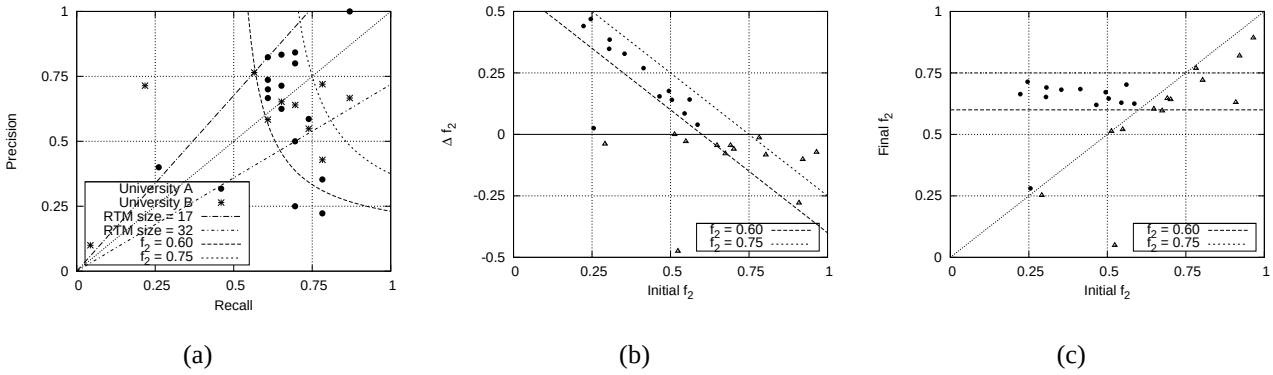
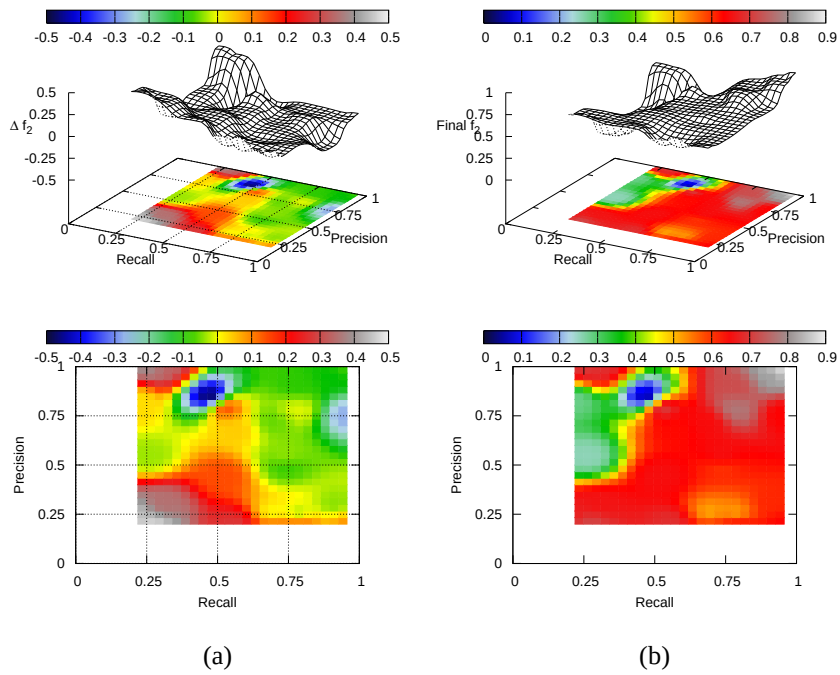


Figure 5. (a) Final candidate RTM distribution; (b) change in f_2-measure versus the initial f_2-measure; (c) final f_2-measure versus the initial f_2-measure.

Recall-Precision Drift: Fig. 3 presents an overview of our results. Each vector plotted on the graph represents the performance of a single respondent with respect to the two main measures of RTM accuracy, recall and precision. The starting point of each vector, marked as a solid dot for participants from one university and as a star for participants from the other, represents the recall and precision of the starting candidate RTM that the study participant received. The end point of the vector shows the recall and precision of the candidate RTM submitted by the participant. Fig. 4 (c) through (f) breaks Fig. 3 by region (quadrant) of the starting candidate RTM. Fig. 5 (a) shows the distribution of the recall and precision of all submitted candidate RTMs.

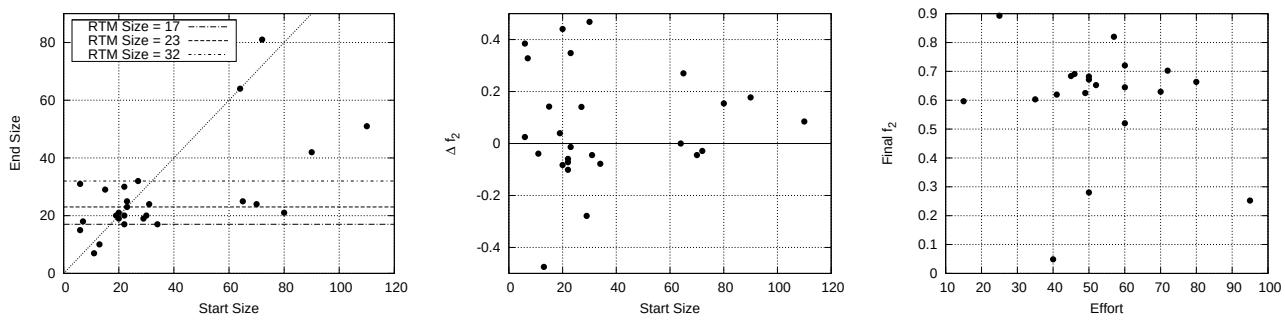
Improving Precision and Recall: In our study, 10 participants improved the recall of their candidate RTM, four kept it the same, and 12 lowered it. Fourteen participants improved the precision of their candidate RTM, two participants kept the same precision, and 10 decreased it. Only seven participants improved both precision and recall. Four participants decreased both.

Who Improved Their Candidate RTMs?: One way of determining whether the submitted candidate RTM is "better" than the starting one is to use the f-measure. In our study, we use the f_2-measure, which prefers improvement in recall (finding all correct links) over improvement in precision (not producing false positives). Fig. 4 (a) and (b)



(a) (b)

Figure 6. (a) Change in f_2-measure vs. initial recall and precision; (b) final f_2-measure vs. initial recall and precision.



(a) (b) (c)

Figure 7. (a) Final RTM size vs. initial RTM size; (b) change in f_2-measure vs. starting size, and (c) final f_2-measure vs. the effort.

show the results for those participants who improved the f_2-measure and those who reduced it, respectively. Overall, 13 participants improved f_2 and 13 did not.

Fig. 5 (b) plots the difference in f_2 values between the starting and the ending candidate RTMs (denoted as Δf_2) vs. the initial value of the f_2-measure. Solid dots represent participants who improved f_2 ($\Delta f_2 > 0$). Triangles represent participants who did not improve f_2. Fig. 5 (c) plots the final values of f_2 vs. the starting values of f_2.

Changes by Region: Fig. 6 (a) and (b) show 3D plots (top) and heat maps (bottom) for Δf_2 and final f_2, respectively. The plots are in the *recall-precision* space.

RTM size: Fig. 7 (a) and (b) show the scatter plots comparing the size of the final candidate RTM to the size of the initial candidate RTM and showing the change in the f_2-measure based on the size of the initial candidate RTM, respectively.

Effort: Fig. 7 (c) shows the f_2-measure of the final candidate RTM plotted against the effort, represented here and elsewhere as the number of minutes to complete the experiment task. Fig. 8 plots the change in the f_2-measure vs. the effort.

B. Analysis

Based on the results described briefly in Section IV-A, we make the following observations concerning the results of our study.

1) Movement Toward $recall = precision$ Line: One of the surprising observations made in the prior study [12] was

the fact that human analysts tended to move their candidate RTMs toward the $recall = precision$ line. With only four data points described [12], though, this observation required more significant confirmation. In our current study, we clearly observed the same drift.

Fig. 5 (a) shows a scatter plot of the *final destinations*, i.e., the locations of the submitted candidate RTMs in the recall-precision space. As can be seen from this figure, in addition to drifting toward the $recall = precision$ line, we observe a hot spot for the final destinations where f_2 is between 0.60 and 0.75 and the size of the RTM is between 17 and 32.

Why do so many final RTMs seem to hover around the $recall = precision$ line? One observation we make is that candidate RTMs with $recall \approx precision$ will have about the same size as the true RTM. This is easily proven: setting equations (1) and (2) equal to each other yields $\sum R_q = \sum n_q$. In other words, the number of links in a candidate RTM is equal to the number of true links when $recall = precision$.

We make a conjecture that analysts have an intuition about the expected size of the true RTM based on the sizes of the artifacts that they are tracing. For example, in the experimental dataset used in this study, one would probably expect the true RTM to contain roughly between 17 and 32 links. If each system test covers exactly one requirement, then there would be 17 links. There would be 32 links if each requirement is satisfied by exactly one system test. As shown in Fig. 7 (a), 19 out of 26 participants submitted RTMs containing between 17 and 32 links. We plan to test this conjecture in our future studies.

2) Regional Behavior Differs: We observe distinctly different results for participants who started with candidate RTMs in different regions.

Participants working with candidate RTMs from the low precision, low recall region (Fig. 4 (d)) drastically improved both precision and recall, and in general, demonstrated the highest improvement in the accuracy of the final candidate RTM. This is also seen in Fig. 6 (a) where a hot spot (gray and bright orange) can be noted in the change in f_2 for participants with starting RTMs in this region.

Participants working with candidate RTMs from the low precision, high recall region (Fig. 4 (f)) improved, sometimes significantly, the precision of the submitted RTM. Changes in recall in this region were generally minor, whether positive or negative. Because low precision, high recall candidate RTMs tend to have significantly more candidate links than the true RTM, we conjecture that the study participants working with such candidate RTMs concentrated mostly on determining errors of commission (false positives) and on weeding them out while not spending too much time trying to find errors of omission (links not in the candidate RTM).

Participants working with candidate RTMs from the high precision, low recall region (Fig. 4 (c)) tended to improve recall, sometimes significantly (with the exception of one case, which appears to us to be an outlier). At the same time, every participant decreased the precision of their submitted candidate RTM. We conjecture that these participants were doing the opposite of what was done in the case of low precision, high recall starting points. Indeed, high precision, low recall candidate RTMs have very few links, so we think that the participants working with these candidate RTMs spent most of their time looking for errors of omission (and introducing both true links and false positives into their candidate RTMs).

Finally, participants working with high precision, high recall (Fig. 4 (e)) candidate RTMs almost uniformly decreased the accuracy (as measured by f_2) of their candidate RTMs. These participants increased neither recall nor precision, but generally decreased both. Most of the decreases were not significant though.

Note that with the exception of a portion of the low recall, high precision starting region, final f_2 was in the range of 0.45 to 0.9 for all starting RTMs (Fig. 6 (b)), with much of the recall-precision space being colored dark orange (0.65) and hotter.

3) Those Who Improved Accuracy: There were 13 participants who improved the overall accuracy (measured as f_2) of their candidate RTM. As Fig. 5 (b) and (c) show, all but one improvement led to candidate RTMs with an f_2 measure value between 0.6 and 0.75. Fig. 5 (c) shows a clear correlation: the lower the f_2 of the initial candidate RTM, the higher was the change in the f_2 measure for these participants.

4) Those Who Did Not Improve Accuracy: On the other hand, the results of the 13 participants who did not improve the RTM accuracy shows a distinctly different pattern. All but two participants showed only a slight decrease in the accuracy of their final candidate RTM (a decrease of 0 to 15%), and this decrease *did not depend* on the accuracy (f_2) of the initial candidate RTM.

5) What is Explained by Effort: It is reasonable to assume that participants who applied minimal effort to the experiment task would yield minimal, if any, improvements in the quality of the RTM. That, however, was not the case. As can be seen in Fig. 7 (c), most all participants submitted an RTM with a final f_2 between 0.6 and 0.75 (only 5 of the 26 fell outside of this range). The effort applied by the participants varied greatly from 15 to 95 minutes. The participant who expended the most effort returned an RTM with f_2 of only 0.25. The participant who applied the least effort returned an RTM with a final f_2 of 0.6. From Fig. 7 (c), we observe a cluster of participants who spent anywhere from 35 to 80 minutes on the task and achieved final f_2 of 0.6 to 0.825. Change in f_2 is even more telling (see Fig. 8). Participants who applied 40 to 70 minutes of effort yielded -0.1 to 0.38 change to f_2 (the participant with -0.5 f_2 is believed to be an outlier). We conclude, therefore, that there is no visible correlation between the effort applied and the final f_2 or the

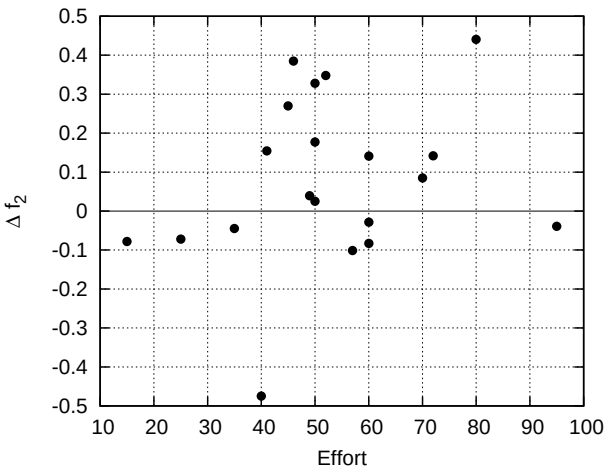


Figure 8. Change in f_2-measure versus effort.

change in f_2.

6) Cal Poly Participants vs. Kentucky Participants: The study was conducted at two universities. Fig. 3 uses solid dots for participants from one university and stars for participants from the other. Overall, we notice that students from the university marked with stars performed slightly below the average in terms of improving recall, precision, and f_2, while students from the other university performed slightly above the average. However, the differences are not large, and the number of participants from each university is not sufficient for us to draw any statistical conclusions. In fact, we have no evidence that suggests that student performance was affected by the university in which they reside. In each of the regions, participants from different universities with similar starting RTMs showed similar tendencies. Each university yielded one outlier in our study: one participant drastically decreased precision and recall of the returned RTM, while another submitted the starting RTM without performing any tracing.

The only interesting observed discrepancy occurred for those who received high precision and high recall candidate RTMs (both recall and precision over 80%). The three participants from one university showed the preference for keeping precision (almost) the same, while decreasing, sometimes significantly, the recall. The two participants from the other university decreased precision significantly more than they decreased recall. It appears that, in this case, participants from one of the universities attempted to concentrate on elimination of false positives (and eliminated some true links), while the participants from the other university attempted to concentrate on discovery of omitted links (and introduced a number of false positives). In our subsequent studies, we intend to see if this pattern holds.

V. Threats to Validity

Influences that may impact the independent variables with respect to causality are referred to as threats to internal validity [17]. A possible personal bias threat in preparation of the golden standard RTM was reduced by having multiple researchers review the answer set. Personal bias in conducting the study was reduced by using a random assignment of students to RTMs. The tool used in the study is also an internal threat. Another possible threat to internal validity is that we kept the strongest matches in the candidate RTMs as we were building RTMs for the participants. Results may have differed had we kept weak matches instead. However, our study tries to mimic what happens when humans observe computer-generated results which do the same thing: show strongest matches first.

There were minimal threats to construct validity as standard IR measures (recall, precision, f_2) were used. These measures have been used extensively in requirements tracing studies. It should be noted that there are other ways and metrics to capture the impact of the independent variable. We have used a subset of those measures.

External threats to validity impact the generalizability of results. In the study, only one experimental dataset was used. The dataset that was used was from a small Java code formatter software project. As this project was developed by upper-division computer science students and may not be representative of a program written by industrial professionals, it is unknown if the results will generalize to other software systems, other software domains, or larger systems. A subset of the Java code formatter program was selected for tracing, to permit completion of the assignment. It is possible that a different group of researchers may extract a different subset of the requirements and test cases, which may lead to different results.

Reliability threats to validity have been mitigated. The study process is defined and repeatable: the study was undertaken at two universities. The second university performing the study had no difficulty applying the study artifacts used earlier by the first university.

VI. Conclusions and Future Work

This paper introduces a simple, repeatable, and adaptable framework for the study of analyst interaction with artifacts generated automatically during the tracing process and describes the initial study conducted at two universities. To our knowledge, this is the first systematic study of human analysts and their impact on the tracing process and its results. In our view, this study confirms the key conjecture of prior studies [12], [13]: *there is a clear need to study this interaction in order to understand how best to automate the tracing process!* At the same time, observed behavior of analysts lends itself to further study.

Based on the results described above, we observe that analysts working with high-quality candidate RTMs do not

necessarily perform better than analysts who start with lower-accuracy candidate RTMs. We saw significant differences in the results of analysts who worked with candidate RTMs from different *recall-precision* regions. Some of the observed behavior leads us to make a number of conjectures about the nature of analyst behavior. Our first conjecture is that *software engineers use sizes of the traced artifacts to estimate the size of the true RTM*. We also conjecture that large, high recall and low precision candidate RTMs make software engineers concentrate on catching errors of commission. At the same time, small candidate RTMs with low recall and high precision make software engineers primarily search for errors of omission.

We plan to address these conjectures in the followup studies we will pursue. We will modify our information collection mechanisms to learn more about the actual tracing process (which, for the purpose of the initial study, was essentially treated as a *black box*).

The overarching goal of our study is to determine which factors influence the work of a software engineer with automated tracing tools. In this paper, we concentrated mainly on measuring the quality of starting and ending candidate RTMs for each participant. We also looked at the effort expanded by our study participants. At the same time, factors outside these (for example, the experience of a study participant), may influence their work. We plan to address this in future studies.

ACKNOWLEDGMENTS

This work is funded in part by the National Science Foundation under NSF grant CCF-0811140. The authors would like to thank John Dalbey for providing us the materials for the Java code formatter dataset. We would also like to thank David Janzen and Gene Fisher for allowing us to run the study in their classes.

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*, vol. 28, no. 10, pp. 970–983, Oct 2002.

[2] 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*. IEEE, May 2003, pp. 125–135.

[3] J. Hayes, A. Dekhtyar, and S. Sundaram, "Advancing candidate link generation for requirements tracing: the study of methods," *IEEE Transactions on Software Engineering.*, vol. 32, no. 1, pp. 4–19, Jan. 2006.

[4] X. Zou, R. Settimi, and J. Cleland-Huang, "Phrasing in dynamic requirements trace retrieva," in *Proceedings of Computer Software and Applications Conference, 2006. COMPSAC '06. 30th Annual International*, vol. 1, Sep 2006, pp. 265–272.

[5] C. Duan and J. Cleland-Huang, "Clustering support for automated tracing," in *Twenty-Second IEEE/ACM International Conference on Automated Software Engieering*. New York, NY, USA: ACM, November 2007.

[6] O. Gotel and C. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of the First International Conference on Requirements Engineering, 1994*. IEEE, Apr. 1994, pp. 94–101.

[7] J. Hayes, A. Dekhtyar, S. Sundaram, and S. Howard, "Helping analysts trace requirements: An objective look," in *Proceedings of the 12th IEEE International Requirements Engineering Conference, 2004*. IEEE, Sept. 2004, pp. 249–259.

[8] J. Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval," in *Proceedings of the 11th IEEE International Requirements Engineering Conference, 2003*. IEEE, Sept. 2003, pp. 138–147.

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

[10] Level 1A (L1A) and Geolocation Processing Software Requirements Specification, *SDST-059A, GSFC SBRS*, September 11, 1997.

[11] MDP Website, CM-1 Project, http://mdp.ivv.nasa.gov/mdp_glossary.html#CM1.

[12] 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*. New York, NY, USA: ACM, 2005, pp. 1–5.

[13] 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*. New York, NY, USA: ACM, 2005, pp. 20–23.

[14] 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*. Washington, DC, USA: IEEE Computer Society, 2007, p. 4.

[15] 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*, vol. 3, no. 3, pp. 193–202, Sep 2007.

[16] S. Yadla, J. H. Hayes, and A. Dekhtyar, "Tracing requirements to defect reports," *Innovations in Systems and Software Engineering: A NASA Journal*, vol. 1, no. 2, pp. 116–124, Sep 2005.

[17] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, and A. Wesslon, *Experimentation in Software Engineering - An Introduction*. Kluwer Academic Publishers, 2000.