2018 IEEE 26th International Requirements Engineering Conference

Vetting automatically generated trace links: what information is useful to human analysts?

Salome Maro*,Jan-Philipp Steghöfer*, Jane Huffman Hayes‡, Jane Cleland-Huang†, Miroslaw Staron*
*Chalmers | University of Gothenburg, Sweden
salome.maro@gu.se, *jan-philipp.steghofer@gu.se*,*miroslaw.staron@cse.gu.se*
†University of Notre Dame, South Bend, IN, USA
JaneClelandHuang@nd.edu
‡University of Kentucky, Lexington, Kentucky, USA
hayes@cs.uky.edu

***Abstract*—Automated traceability has been investigated for over a decade with promising results. However, a human analyst is needed to vet the generated trace links to ensure their quality. The process of vetting trace links is not trivial and while previous studies have analyzed the performance of the human analyst, they have not focused on the analyst's information needs. The aim of this study is to investigate what context information the human analyst needs. We used design science research, in which we conducted interviews with ten practitioners in the traceability area to understand the information needed by human analysts. We then compared the information collected from the interviews with existing literature. We created a prototype tool that presents this information to the human analyst. To further understand the role of context information, we conducted a controlled experiment with 33 participants. Our interviews reveal that human analysts need information from three different sources: 1) from the artifacts connected by the link, 2) from the traceability information model, and 3) from the tracing algorithm. The experiment results show that the content of the connected artifacts is more useful to the analyst than the contextual information of the artifacts.**

I. Introduction

Traceability is regarded as important in software and systems engineering [1], [2]; however, its adoption across many industrial sectors is still quite low [3]. One of the main inhibitors to adoption is the cost of creating and maintaining traceability links [4], [5]. To combat this challenge, many automated techniques for creating and maintaining trace links have been proposed [6] including techniques based on information retrieval [7], [8], machine learning [9], deep learning [10], rule-based [11], and repository mining [12]. Automated techniques are promising, since they could potentially eliminate the manual work of creating and updating trace links. However, current solutions do not yield perfect results in terms of either precision or recall [8], [13]. Human analysts therefore need to inspect the generated trace links and make a final decision on their correctness. We refer to the task of inspecting automatically generated links to confirm true links and reject false links as "vetting." The task of vetting trace links is tedious, and it has been observed that, in some cases, instead of improving the accuracy of the generated trace links, human analysts actually decrease their quality [14].

Researchers have conducted two types of studies on the traceability vetting process: studies that investigate the impact of varying the accuracy of the generated traceability model and studies that build and evaluate tools to support analysts. Regarding the former line of research, Cuddeback et al. [14] showed that while human analysts tend to improve the trace model if it has a low precision and recall, they tend to degrade the quality of the trace model if it initially has high precision and recall. Regarding the latter line of research, Hayes et al. [15], e.g., developed RETRO, a traceability management tool that generates links automatically and presents them to a human analyst for vetting. The tool offers features such as global tracing, local tracing, and filtering the generated links based on a score from the tracing algorithm. The main focus of such studies is the usability of the tool and which benefits for vetting trace links it provides over a manual tracing approach. However, further empirical studies are needed to understand which factors influence human analysts' decisions when vetting trace links in order to improve this process [16], [17] within specific software engineering contexts [18].

In this study, we take a different perspective and hypothesize that the task of vetting automatically generated links can be improved if the traceability tool provides useful information to the human analysts. We specifically investigate how *context information* can be useful to the human analyst. In our study, we use the definition of context information from Abowd et al. [19]: context information refers to any information that can be used to characterize the development artifact – e.g., the meta data of an artifact, such as the date it was created or modified. In contrast, the content of the artifacts, such as the code in a Java file or the textual description of a requirement, is not considered context information. Our assumption is that offering context information together with the content of the artifacts will improve the decisions made by human analysts.

Our study has two main contributions. First, we conduct an empirical investigation investigating what context information is needed by human analysts. We collect data from practitioners and compare it to existing literature. Second, we investigate the effect of context information for supporting the human analyst during the vetting task. The study addresses the following research questions:

2332-6441/18/$31.00 ©2018 IEEE
DOI 10.1109/RE.2018.00-52

RE 2018, Banff, Canada
Research Paper

RQ 1: What context information is useful to human analysts when vetting trace links?

RQ 2: To what extent does this information help analysts to make correct decisions?

The remainder of the paper is structured as follows. Section II provides an overview of the related work in the area of vetting trace links while Section III describes our research methodology. Our results are described in Sections IV, V, VI, and VII, while Section VIII discusses the results with respect to existing research. We conclude with a discussion of threats to validity in Section IX and then summarize our findings in Section X.

II. Related Work

Related work primarily falls under the two areas of vetting trace links and tools for supporting human analysts in the vetting task. We discuss each of these areas in this section.

A. Performance of human analysts in vetting trace links

Several studies confirm that human analysts make mistakes when vetting automatically generated trace links. Cuddeback et al. [14] studied the performance of human analysts in evaluating the correctness of generated trace links. They showed that humans often degrade the accuracy of generated links by accepting wrong links and rejecting correct links. As a result of these findings, researchers have studied factors that contribute to human analysts' decision making in order to design tools to improve the quality of human-vetted links.

Dekhtyar et al. [20] investigated several factors that may influence the accuracy of trace links produced as a result of human decision making. They showed that the quality of the initial trace link set and the effort in analyzing the links both made a difference in the final accuracy. Sets of trace links that started with high recall and precision tended to decrease in both recall and precision, while sets that started with low recall and low precision led to improvements in both recall and precision. Similarly, a starting set with a high precision and low recall led to improvements in recall but decreases in precision, and vice versa. They also observed that analysts who reported investing more effort in vetting the links often ended up reducing recall by rejecting true links. These findings align with those from similar experiments reported by Cuddeback et al. [14] and Kong et al. [21]. One possible explanation for their results is the presence of gray links, i.e., those links which capture meaningful associations between artifacts, but are ambiguous because they are only relevant for certain software engineering tasks [18].

Additionally, Kong et al. [22], studied how humans make correct and incorrect decisions by investigating logs recorded during the vetting process. They identified several strategies that analysts use to make decisions such as "first good link" (participants focusing on finding the first good link) and "accept-focused" (where participants only accepted links and never explicitly rejected links). This study also showed how the tracing experience of the analyst and effort spent on vetting traces affects the quality of the final set of traceability links.

Table I
Interview Subjects

Subject	Role	Domain
A	Business Analyst	Finance
B	Software Architect	Automotive
C	Practice area lead, software and license handling	Telecommunication
D	Software Lead	Military
E	Requirements engineering researcher	IT Consultancy
F	Research Fellow	Automotive
G	System Engineer	Software Development
H	Requirements engineering researcher	IT Consultancy
I	Verification and Validation analyst	Space
J	Senior Software Architect	Automotive

Another study reported by Dekhtyar and Hilton [23] investigated the strength and weaknesses of human analysis versus automated tracing techniques in order to identify how the two approaches could best complement each other. They proposed using an automated technique that identifies links which human analysts would be more likely to miss.

Our study, however, takes a different approach by studying context information that can be presented to the human analyst in order to make the decision process easier and more accurate. Kong et al. [21] already observed that during the vetting task some analysts go back to review the connected artifacts and other artifacts in the data set before they decide on a link. This implies that the analysts seek to understand the artifacts and how they fit together in the data set in order to make decisions.

B. Tools for vetting trace links

There are several research tools that support the vetting of trace links. It is important to note that not all tools that can generate trace links offer functionality for vetting those links. Many research tools stop at the generation stage (see, e.g., [24]), because they are focused on presenting or improving certain automated techniques to generate the links, and are not interested in further steps such as vetting or utilizing links. The main functionalities provided by tools that support link vetting are: generating links, presenting them to the analyst, and finally allowing the analyst to accept correct links and to reject incorrect links. Some tools also allow the analyst to search for missing links [15], [25] and to perform coverage analysis for completeness of links [26]. To the best of our knowledge, there are six tools that support vetting of trace links: RETRO [15], ADAMS [27], Poirot [28], TraCter [29], TraceME [26], and ART-Assist [30]. A deeper analysis of these tools is presented in Section V.

III. Research Method

The study was conducted using the design science research method [31], [32] in which a problem is iteratively investigated while implementing suitable artifacts to solve the problem and evaluating the effectiveness of the solution. We utilized a combination of techniques based on interviews, literature review, and a series of controlled experiments. The process was conducted in three distinct iterations.

A. Interviews and Identification of Existing Tools

In the first iteration, we conducted a series of semi-structured interviews to investigate what information affects human analysts' decisions when vetting traceability links. We first designed an interview guide[1], for which the questions were reviewed and re-written by four researchers across several iterations. We then tested this interview guide by conducting one pilot interview in order to establish its soundness and to improve the guide. After the pilot, we interviewed ten practitioners with experience in creating and maintaining traceability links. We used convenience sampling where our aim was to interview practitioners with experience in traceability while maximizing diversity in our interview sample. All the interviewees were recruited through our personal connections and were from different domains. Table I provides a summary of our interviewees, their roles, and the domains in which they work. The interviews focused on identifying which types of information analysts used to create traceability links and how trace links are evaluated in the interviewee's company. We assume that this same information should also be available during the process of vetting automatically generated trace links. In addition, we asked the interviewees about their experience with automatic tools for traceability link generation and how they would expect such tools to work. For each interview, we summarized a set of information and information sources discussed by the interviewee in a spreadsheet.

To further strengthen our results, in the second iteration of our study, we researched existing literature on tools that support the vetting of trace links that have already been published in scientific literature. Our aim was to identify if these tools already offer the information collected from the interviews and to identify any gaps that might exist. We conducted our literature search by starting with previously identified papers on trace link vetting and then using snowballing to extend the scope. Many of these papers are already discussed in Section II describing related work.

B. Experimental setup

To validate the usefulness of context information during the vetting process, we created a prototype by extending an open source traceability management tool, Eclipse Capra [33]. We chose to extend Eclipse Capra for this study because it is a customizable open source tool that contains basic traceability features such as creating and visualizing trace links. Additionally, two of the authors are developers of the tool. Details of how Eclipse Capra was extended are provided in Section VI. Using the prototype we conducted an experiment with 33 participants in order to understand how various types of context information affects the human analyst's decision making process.

Experiment Variables: The experiment was designed to investigate whether providing analysts with contextual information from connected artifacts would affect their performance in vetting trace links. Examples of contextual data include meta data (e.g., date created), attributes (e.g., status or priority), location (e.g., subsystem), and connectivity to other artifacts. The independent variable was therefore context information of the artifacts, where one group was given contextual information and the other was not. The choice of contextual information to include in our study was driven by results from our interviews and literature review as reported in Section IV. We measured three dependent variables: 1) recall of the final trace links; 2) precision of the final trace links; and finally 3) the number of links investigated by the analyst within the allotted time.

To reduce the number of confounding factors, we controlled three variables: 1) the initial precision of the trace links, 2) the initial recall of the trace links, and 3) the order in which the experiment subjects vetted the trace links. All analysts were instructed to review the provided list of links from top to bottom as provided to ensure that they inspected the same links without intentionally skipping any links. This also made it possible for us to identify links investigated by the analysts for which they did not indicate a decision.

In the experiment, an analyst who produced trace links with higher final recall, higher final precision, and who investigated more links was considered to have outperformed an analyst with lower final recall, lower final precision, and a lower number of investigated links.

Based on our research questions, we formulated the following hypotheses to investigate the three dependent variables:

H_{o1} Providing human analysts with context information about the artifacts connected by trace links has no effect on the precision of the final set of trace links

H_{o2} Providing human analysts with context information about the artifacts connected by trace links has no effect on the recall of the final set of trace links

H_{o3} Providing human analysts with context information about the artifacts connected by trace links has no effect on the number of links investigated during a given time period

These hypotheses guided our evaluation into how and to what extent context information is useful to the analyst (RQ2).

C. Experiment Materials

We describe the experiment artifacts that the candidates interacted with as well as the data collection instruments.

Experiment Artifacts: We used Medfleet, a system developed by Software Engineering graduate students as part of a five month studio course [34]. We selected this system because it contains realistic artifacts of a software development project, and has been used in a previous publication [35] with a set of verified traceability links. Medfleet enables its users to request emergency medical kits to be delivered using small Unmanned Aerial Systems. The project artifacts include requirements, environmental assumptions, architectural components, code, and fault descriptions. Requirements, assumptions, and faults were originally captured in Jira, while code was written in both Java and Python and stored in GitHub. The trace links provided by the project served as a gold standard and consisted of 288 true links. In the experiment we used a subset of the artifacts – requirements, Java code implementing the mission

[1]https://tinyurl.com/yauypdhn

control subsystem, assumptions, and faults to reduce the scope and make the experiment more manageable. We used the Vector Space Model with Term Frequency-Inverse Document Frequency (TF-IDF) technique to generate three types of traceability links: links from requirements to code, links from requirements to assumptions, and links from requirements to faults. TF-IDF is an information retrieval technique that not only uses text similarity to predict how similar two artifacts are, but also uses term frequency [36]. The weight of the terms is calculated as a product of the frequency of the term in a given document and the inverse of the frequency of the term in all the documents. This technique gives an indication of how important a term is in a given document. The total number of links generated was 1239 with a precision of 42% and recall of 33%.

Data Collection Instruments: We created two questionnaires to collect data from our experiment: 1) a pre-experiment questionnaire that collected information about the participants' experience with software development, use of the Eclipse IDE (on which Eclipse Capra is based), and experience with traceability, and 2) a post-experiment questionnaire that collected feedback on the experiment and the different features of the tool with which users interacted. Additionally, we recorded the screen during the experiment. To collect information about the vetted links, we stored all the links that the analyst accepted in a list of accepted links and all the links that the analyst rejected in a list of rejected links. These lists were stored as EMF models in Eclipse Capra.

Experiment Subjects: We used convenience sampling to identify diverse participants from our personal connections. As a result, the experiment was conducted with 33 participants of which six were Bachelor students, eight were Masters students, twelve were PhD students, and seven were industry practitioners. The students were all software engineering students from two universities and therefore had all taken several courses on software development. The subjects were randomly divided into two groups, the control group (16 subjects) and the experiment group (17 subjects). Out of the 33 subjects who took part in the experiment, we discarded the results of five subjects because they did not follow the instructions of the experiment, for example, by evaluating trace links in a different order from the instructions or evaluating only one type of trace links. These results were excluded in order to avoid bias.

Experiment Groups: The control group was provided with a version of the tool which did not display context information, while the experiment group was provided with a version of the tool with context information of the connected artifacts. This means that the control group had features F1 to F8 and the experiment group had features F1 to F8 and additionally features F9 and F10 (cf. Section VI).

Experiment Procedure: All experimental sessions began with one of the researchers giving a scripted brief introduction to traceability and automated techniques of generating trace links. This was followed by explaining the Medfleet system and the vetting task to the participants. During this session, participants were allowed to ask questions. The instructions of the experiment were also distributed in paper format for participants to read [2]. Before the experiment, participants filled in the pre-experiment questionnaire. The participants were then given 45 minutes to vet as many links as they could. At the end of the experiment, we collected the final traceability models the participants produced as well as video recordings of the screen for the entire 45 minute vetting session. The participants also filled in a post-experiment questionnaire containing questions about the task and which features of the tool they found useful. The questionnaires for each group are available online [3][4]

IV. Interview Results

From the interviews, we learned that the task of vetting trace links is conducted in companies, even when trace links are created manually. In safety-critical domains, trace links must be carefully reviewed before submission to the certification body [37] . The information used to evaluate the correctness of a trace link is therefore used during both the link creation and the link assessment processes. As a result, information collected from the interviews was derived from two activities, that of creating and reviewing the links. We categorized the information that the interviewees reported into three main categories of information derived from 1) connected artifacts (Section IV-A), 2) the traceability information model showing connections between artifact types (Section IV-B), and 3) results from the tracing algorithm (Section IV-C). Additionally, interviewees reported how they would like this type of information to be represented or displayed (Section IV-D). From these results we ultimately selected specific contextual and content-based elements to be included in our tool and used in the evaluation.

A. Information from the connected artifacts

Six out of ten interviewees reported that they create trace links based on their knowledge of the system. They use their experience to determine if two artifacts (e.g., a specific requirement and a specific Java class) should be connected or not. However, when asked what information they would need if they did not have such system experience, they reported the following:

- The content of the connected artifacts: this refers to the information that makes up the artifacts, e.g., the content of the Java file represented by the actual lines of code, or content of a requirement represented by its textual description.
- The meta data of the connected artifacts, such as who created it, when it was modified, and who modified it.
- Other artifacts connected to the artifacts: This refers to other development artifacts that already have established links with the investigated artifacts, e.g., when deciding if requirement X is connected to a Java file Y, one may

[2]https://tinyurl.com/y98jpdtg
[3]https://goo.gl/forms/zdY23Gqjk1rixF4U2
[4]https://goo.gl/forms/jMRKW9yWitHDj4JI3

first want to know which other requirements are related to requirement X or to the Java file Y.
- The location of the artifacts in a project, system, or subsystem.

B. Information from the traceability information model

Before creating or reviewing trace links, all the interviewees reported that it is necessary to understand how the different artifacts in the project are related to each other. For instance, interviewee E reported that the company has a metamodel that specifies how the artifacts should be connected. Such a metamodel or traceability information model (TIM) contains information indicating specifically which types of artifacts may be linked together within a given project. For example, a link might be allowed from an acceptance test to a requirement, but not directly from the acceptance test to code. In summary, users need the following information from the TIM:

- A definition of which links are allowed and which are disallowed. The TIM may also contain additional information about cardinality constraints.
- The type of link that is being created or reviewed. For TIMs that contain diverse link types (e.g., tests, refines, describes), the analysts need to understand the type of link they are currently vetting before making a decision.

C. Information from the tracing algorithm

While many interviewees were aware of techniques for automatically generating trace links, only three had actually used such tools and had first-hand experience of tracing algorithms. For interviewees that had no experience with these tools, the interviewer carefully described how they worked. We then asked all participants what information they would expect to see if they used an automated tool to create links. This was first asked as an open question, and if the interviewees did not have any ideas, we suggested options. All the interviewees agreed that a confidence score for each link would be useful. Additionally, some of the interviewees agreed that for information retrieval techniques that use text matching, seeing which exact words or phrases from the source artifacts matched words or phrases in the target artifact would be beneficial. Therefore we concluded that the following information should be available for evaluating trace links:

- A score representing the similarity of two artifacts. This score is calculated by the information retrieval algorithm that is used to generate the links.
- Words or phrases from the source artifact that matched words or phrases in the target artifact.

D. Presentation of the information

Especially for large projects, the number of artifacts and trace links can be overwhelming and therefore tool support is critical for creating and reviewing links. Our interviewees reported that when reviewing trace links, they would like to search for artifacts and filter the links that they are reviewing. Additionally, it was reported that being able to review one type of trace link (e.g., requirements to code) is beneficial, compared to viewing all the trace link types at once.

Two out of ten interviewees reported that they would like to have a graphical representation of the trace links. However, they noted that since a large amount of trace links can lead to large graphs which are complex to read, the traceability management tool should allow filtering of the links to display manageable graphs.

From this category, we identified two features that should be included:

- Ability to search for and filter trace links.
- Ability to view trace links in a graphical representation.

Table II
CONTEXT INFORMATION

Context Information	**Description**	**Example** (w.r.t. Requirement (RQ-01))
Meta data	This refers to data describing the artifact e.g., who created the artifact, when it was created, when it was modified	created on: May 19, 2016, createdBy: SMaro
Location	Where the artifact is located and in which system	MedFleet/Requirements.xlxs
Connected artifacts	Other artifacts linked to the artifact in question	Assumption (A-01), Fault (F-01)

E. Context information

Since the interviewees reported generally on what information is useful when vetting links, we collected this information and used our definition of context information (information that can be used to characterize the development artifact) to derive context information relevant for trace links vetting. This information is summarized and exemplified in Table II.

V. INVESTIGATION OF EXISTING TOOLS

We compared the results from the interviews to available literature on vetting traceability links. Specifically, we investigated research tools that support vetting of traceability links. Our literature search started with a few papers that are prominent in the field, e.g., [15], [14], [20] and used snowballing to acquire more papers. We found six tools described in scientific literature that support traceability link generation and the activity of vetting traceability links. These tools and their features are summarized in Table III along with the extended Eclipse Capra. Our investigation was focused on features that provide information to the human analyst and did not focus on analyzing the different tracing strategies provided by the tools. We used this knowledge from the existing tools as inspiration for our own implementation of features that were suggested by the interviewees. The implementation decisions and details are discussed in the next section.

VI. PROTOTYPE IMPLEMENTATION

In order to perform our evaluation using a controlled experiment, our prototype needed to automatically generate trace links given a set of development artifacts and then to present

Table III
INFORMATION PROVIDED BY EXISTING TOOLS THAT SUPPORT TRACE LINKS VETTING. "YES": INFORMATION/FEATURE PRESENT; "NO": INFORMATION/FEATURE NOT PRESENT, "N/A": NOT APPLICABLE; "UNK.": NOT ENOUGH INFORMATION AVAILABLE TO DETERMINE IF FEATURE PRESENT.

Information	RETRO	Poirot	TraceME	AdamsTrace	TraCter	ART-Assist	Eclipse Capra
Navigation to full artifact content	Yes	No	Unk.	Unk.	Yes	Yes	Yes
Score from the tracing algorithm (Similarity measure)	Yes	Yes	Yes	Yes	No	No	Yes
Matching terms from the source in the target artifact	Yes	Yes	No	Yes	No	No	Yes
Text search	Yes	Yes	Unk.	Yes	Unk.	Unk.	No
Trace link type	N/A	No	Yes	Yes	Yes	Yes	Yes
Graphical representations	No	Yes	Yes	No	No	No	Yes
Accepted links	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Location of the artifact	Unk.	No	Unk.	Unk.	Yes	Yes	Yes
Summary of the artifacts (e.g., Java class documentation)	No	No	No	No	No	Yes	No

these links for vetting, together with context information, to the human analyst. To generate the initial set of trace links, we extended Eclipse Capra to use the Vector Space Model – Term Frequency–Inverse Document Frequency (TF-IDF) [36]. This technique is proven useful in previous studies [30].

To provide the ability to vet the generated traceability links, we implemented ten features identified through the interviews. Features 1 to 8 provided basic functionality for vetting links, while features 9 and 10 presented context information. Our choice of what context information to include and evaluate is based on the results of the interviews. Three of our interviewees reported that meta data of the connected artifact was useful, four mentioned location of artifacts (e.g., artifact X and Y are both from the same subsystem or package) and five mentioned relationship to other artifacts. Because our aim was to study whether a type of information, and not its exact representation, was important, we made every effort to select a simple solution for each type of information. We provide the rationale for the way each feature was implemented below and illustrate some of these features in Figure 1.

F1: A list of trace links generated from the tool: We displayed trace links in the form of a tree list where the parent of the tree represents the source artifact and the children of the tree represent the target artifact. This type of display is what is known as local tracing, where the user can view links related to one artifact at a time. We chose this implementation since it was preferred by users in a previous study reported by Hayes et al. [15].

F2: Ability to open the artifacts connected by the links: To make sure that users can access the content of the connected artifacts, we implemented functionality to open the connected artifact for each of the artifacts This feature is present in some of the tools, e.g., in RETRO, TraCter, and ART-Assist. In RETRO, the content of the requirements are displayed as text. In TraCter and ART-Assist, connected Java files are opened in a pop-up window. In our case, we decided to use the native environments of the tools that were used to create the artifacts. For instance, all Java files are opened in a Java editor so that features such as syntax highlighting are available.

F3: A display of similarity scores generated by the tracing algorithms: With the exception of TraCter and ART-Assist, the rest of the tools display the similarity score from the trace generation algorithm. This feature has also been highly ranked by user studies performed on RETRO [15] and ADAMS [7]. There were two possibilities to display this score. One way is to display the raw value (or a percentage) and the other way is to translate this into a confidence value as done in Poirot [28]. We chose to display the values as raw numbers since this has been shown to work and was highly rated [7], [15] while there are no user studies on the alternative by Poirot.

F4: A graphical representation of the links: We implemented a graphical representation of the links that displays trace links as a graph. The source of the trace link was the root node of the graph, while the targets of the trace link were the child nodes. We did not implement any transitive links, i.e., displaying more than one level of traceability, because three of our interviewees had indicated that for the purpose of vetting the trace links, they could only deal with one level of trace links at a time.

F5: A view of the Traceability Information Model (TIM): This feature was requested by all interviewees. Additionally, since we wanted our tool to support diverse link types, displaying the TIM to the user gives information on which links are allowed and the different constraints on the different link types. Several tools such as RETRO, only support one type of trace link, i.e., tracing high level requirements to low level requirements, and therefore displaying the TIM was not a required feature.

F6: Ability to see the link type: This feature is related to feature F5. When vetting the trace links, the user should be able to know which link type they are currently vetting. We implemented this by adding the link type name, e.g., "satisfies," "realizes," or "requirements to code" depending on how the TIM is defined for the project.

F7: Ability to see terms from the source artifacts that matched terms from the target artifacts: This feature enables the user to see which term in the source artifact matched terms in the target artifact. The most common way of implementing this is through highlighting these terms in the source artifact and in the target artifact [15], [28]. In our case, due to technical limitations (it is tricky to implement multiple highlighting in the Java editor),we

Table IV
SOFTWARE DEVELOPMENT EXPERIENCE OF THE EXPERIMENT SUBJECTS

	<6 months	**6-12 months**	**1-3 years**	**3-6 years**	**>6 years**
Experience with software development	2	2	11	5	8
Experience with Eclipse	9	2	9	3	5

followed a simpler solution and used markers to indicate lines where these terms occurred. Since there can be many terms, we only displayed the top three terms according to their TF-IDF weights. This feature was only implemented for Java files.

F8: Ability to accept a link and reject a link: This feature was implemented to allow the analyst to accept and reject links by using the trace link list provided. The analyst could accept/reject one target for a link at a time or could accept/reject all suggested targets at once. This kind of implementation has been shown to work in RETRO and TraceME. When an analyst accepts a link, it is removed from the candidate trace link list and added to a list of confirmed links. When an analyst rejects a trace link as incorrect, the link is removed from the candidate trace links and added to a separate list of rejected links. Storing both the list of accepted links and rejected links enables the tool to keep track of what the analyst has already vetted in order not to show these links to the analyst again for vetting.

F9: **Ability to hover over the connected items and see context information for the artifacts**: Displaying context information (e.g., date created, date modified, who created and modified the artifact, and the location of the file) was a new feature that has not been implemented in existing tools, with the exception of the location of the artifact in ART-Assist. We implemented the display of the context information based on the suggestion by the interviewees and show the context information in a tooltip when the mouse hovers over an artifact.

F10: The ability to see already accepted links: Other links that have already been accepted and contain the artifact that is currently being inspected are additional context information. When analyzing links related to requirement X, e.g., the analyst would like to know if other links to requirement X already exist. While the existing tools that allow the analyst to view accepted links show the entire list, in our case the user can see only accepted links related to the artifacts the user is currently inspecting. If the user is vetting links related to requirement X, then the user can only see accepted links that contain requirement X either as a source or a target. This implementation is due to the fact that, in our case, the accepted links are a form of context information while other tools show accepted links to indicate progress to the user.

VII. EXPERIMENT RESULTS

In this section, we report results for all participants who successfully completed the study (i.e., 14 in the control group, and 14 in the experiment group). The results of the pre-experiment questionnaire are shown in Table IV. Most of our participants had at least one year of experience with both software development and Eclipse. However, only eight of our participants had experience with traceability activities.

We recorded the total number of accepted links, total number of rejected links, number of correct accepts, number of correct rejects, and the total number of links investigated. We also computed the final recall and precision.

As can be seen from the results in Table V, the performance of the two groups was quite similar. Since the data was not normally distributed, we used the Wilcoxon-Mann-Whitney U Test [38] with an alpha of 0.05 to determine if there was a statistically significant difference between the two groups for the following three variables: 1) the recall of the final trace link set, 2) the precision of the final trace link set, and 3) the number of links investigated by the analyst in the given time. The results show that for all three variables, the difference is not statistically different between the two groups. Therefore we cannot reject the null hypotheses (cf. Section III).

To get a better understanding of why there was no significant difference between the two groups, we further analyzed the post-experiment questionnaire results. Each group had a post-experiment survey that asked about the different information provided in the tool and their perception of how useful this information was for the vetting task. Since the experiment group had the contextual information, we analyzed their perception on this extra information. The survey included three questions related to the extra information given to the test group. The first two questions were on the metadata of the connected artifacts. All the questions were to be answered on a 5-point Likert scale, where 1 is strongly disagree, 2 is disagree, 3 is neutral, 4 is agree, and 5 is strongly agree. The first statement was "Knowing who created the connected artifacts was useful when vetting the trace links." For this statement, eight of the respondents said they strongly disagree, three disagreed, only one responded with agree, and two did not answer the question. A similar trend was seen for the second statement which was "Knowing when the connected artifacts were created was useful when vetting the trace links." For this statement, eight respondents stated that they strongly disagree, three stated that they disagreed, while only two agreed to the statement, and one was neutral (cf. Figure 2). We were able to ask some participants why they thought this information was not useful during their vetting task. Most of them responded that since they did not know the system or the people involved in creating the system, this information was not useful. However, they noted that if they were involved in the development of the system and knew the participants, then this information might have been useful. This corresponds to what our interviewees reported, that their experience with the system is important when vetting trace links.

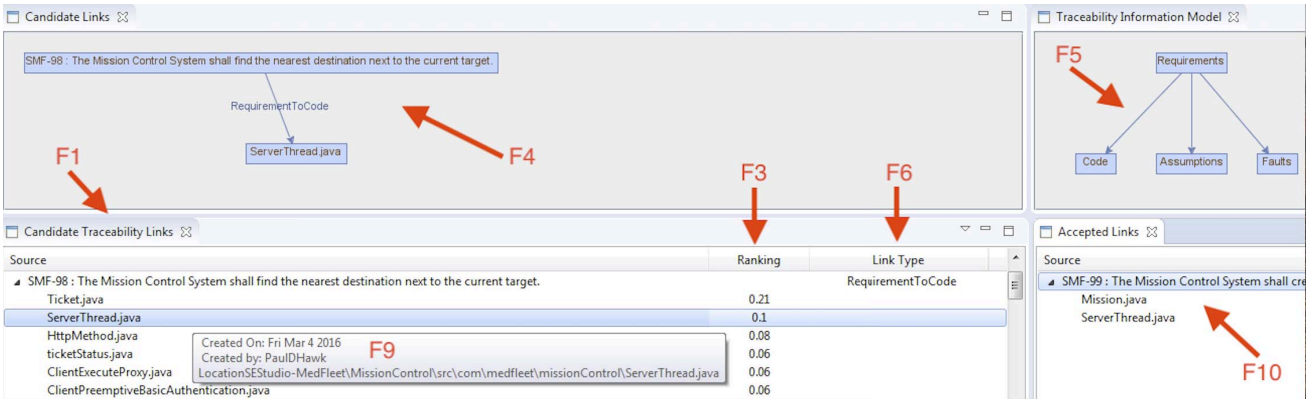

Figure 1. A screen shot of the trace links vetting tool with indicators for the different features.

Figures 2 and 3 show that three features were perceived as most useful during the vetting task: 1) the ability to open the connected artifacts, 2) the ability to know which link type is being vetted, and 3) knowing the similarity score of the link from the algorithm. This information also explains why our null hypotheses could not be rejected since the content of the connected artifacts was considered more useful than contextual information of the connected artifacts.

We conducted further analysis on our data to understand if there was a significant difference between the performance of industry participants and the performance of students. Due to a lower sample size of industry participants (four in the control group and one in the test group), we could only compare this for the control group. For both the precision and total number of links vetted, using the Wilcoxon-Mann-Whitney U Test, there is no statistically significant difference between the performance of the two groups. However, for recall, we found that there is a significant difference between the groups, where the student group had slightly higher recall (see Table VI). Going back to the definition of recall (the number of correct links identified over the total number of correct links present in that set of links vetted), we can see that the students that had higher recall are those that vetted a small number of links. Subject F (in Table VI), e.g., did not reject any links. This therefore does not mean that the students had better performance, only that they vetted a smaller number of links and therefore reduced their chances of rejecting correct links.

Analyzing the post-experiment questionnaire for the industry participants and student participants, we see the same trend in the top three features that were found to be useful during the vetting process: 1) the ability to open the connected artifacts, 2) the ability to know which link type is being vetted, and 3) knowing the similarity score of the link from the algorithm (cf. Figure 4).

VIII. Discussion

In this section, we discuss the results of our study with respect to our research questions. As previously stated, our interviews revealed that three sources of information are useful during trace link vetting: 1) information from the connected artifacts, 2) information from the traceability metamodel, and

Table V
Experiment Results

Test Group						Control Group					
Subject ID	Accepted	Rejected	Investigated	Precision (%)	Recall (%)	Subject ID	Accepted	Rejected	Investigated	Precision (%)	Recall (%)
TS01	100	33	149	13	87	CI01	54	81	135	19	56
TS02	79	85	164	14	61	CS01	49	44	98	20	67
TS03	27	38	66	33	60	CS02	12	2	19	25	100
TS04	60	13	95	20	92	CS03	111	49	160	10	61
TS05	58	39	128	19	92	CS04	17	67	84	47	53
TS06	49	128	177	14	39	CI02	46	92	138	20	50
TS07	130	207	337	13	63	CS05	47	84	134	21	56
TI01	73	61	134	16	67	CS06	103	56	159	16	87
TS08	14	57	71	29	27	CS07	42	0	42	24	100
TS09	37	20	57	32	80	CI03	73	36	109	19	82
TS10	40	55	95	28	69	CS08	210	113	323	11	92
TS11	98	1	99	16	100	CS09	17	19	36	29	56
TS12	82	26	108	16	76	CS10	47	7	54	32	100
TS13	270	53	323	9	96	CSI04	64	120	184	16	50
Average	**79.79**	**58.29**	**143.07**	**19.50**	**72.02**	**Average**	**63.71**	**55**	**119.64**	**22.04**	**72.23**

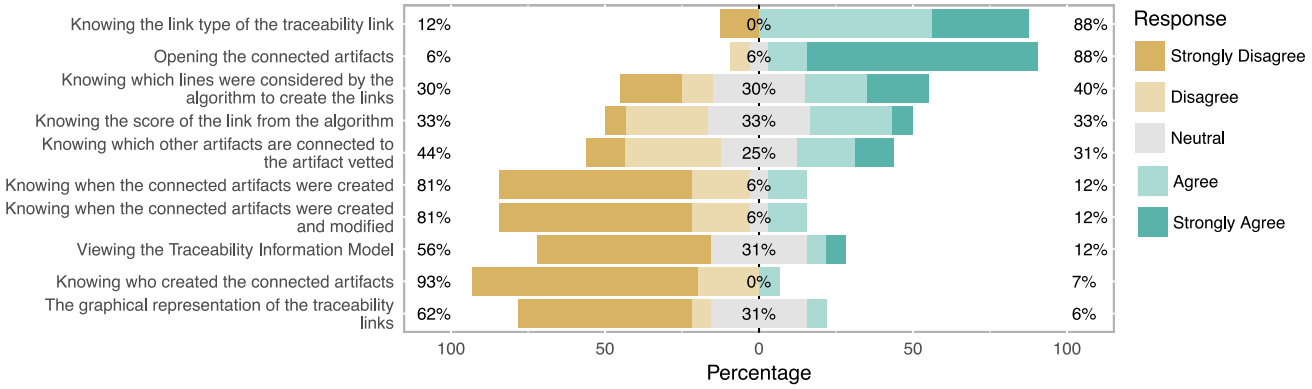

Figure 2. Results of the test group from the post-experiment questionnaire

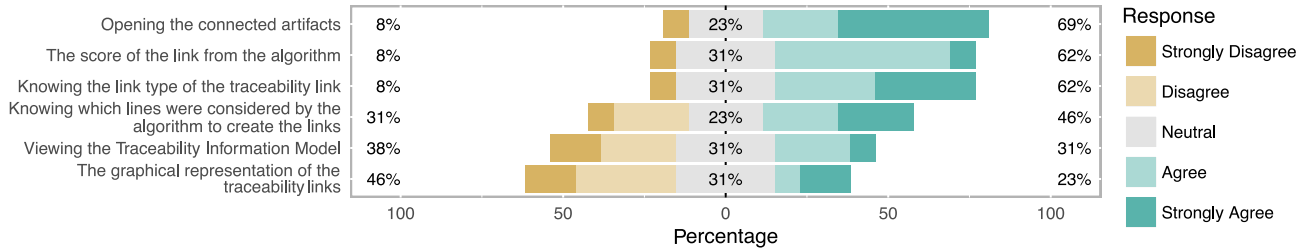

Figure 3. Results of the control group from the post-experiment questionnaire

3) information from the tracing algorithm. We designed an experiment to validate how context information affects the analyst's performance when vetting trace links. The context information we studied came from the first source of information i.e., the connected artifacts. Our experiment showed that there is no statistically significant difference between the group that was provided with context information and the group that was not. The experiment also showed that the analysts spent more time investigating the content of the artifacts than investigating the context information provided. This could be due to unfamiliarity with the system and its context, analysts tried to understand the artifacts by looking at their content rather than their context information. This behaviour of the analyst is also supported by what was reported by the interviewees. A majority reported that familiarity and experience with the system is important when vetting trace links.

Table VI
PERFORMANCE OF INDUSTRY SUBJECTS VS. STUDENT SUBJECTS

Industry Subjects					
Subject	Accepted	Rejected	Investigated	Precision	Recall
A	54	81	135	19	56
B	64	120	184	16	50
C	111	49	160	10	61
D	46	92	138	20	50
Average	**68.75**	**85.5**	**154.25**	**16.25**	**54.25**
Student Subjects					
A	49	44	98	20	67
B	12	2	19	25	100
C	17	67	84	47	53
D	47	84	134	21	56
E	103	56	159	16	89
F	42	0	42	24	100
G	73	36	109	19	82
H	210	113	323	11	92
I	17	19	36	29	56
J	47	7	54	32	100
Average	**61.7**	**42.8**	**105.8**	**24.4**	**79.5**

While there are some differences between industry and student participants (cf. Table VI), the difference is not statistically significant. We believe that it is experience with the particular system that plays a larger role in the performance of the analyst instead of general software development experience. The study by Dehtyar et al. [20] also shows neither experience with software development nor tracing experience had an influence on the analysts' performance. *We therefore suggest that analysts who have system experience be assigned to the task of vetting traceability links*. Developers should, e.g., be assigned to vet links between requirements and code and safety analysts should be assigned to vet links between requirements and faults. However, further research is needed to support this.

Regarding the second source of information, i.e., the traceability information model, participants from both the control and the experiment group thought that knowing the link type of the trace link was useful, but having the entire TIM visible to them at all times was not very useful. This could be because the experiment only included four types of elements (requirements, code, assumptions, and faults) and therefore the TIM was small. The analysts only had to see the TIM once to understand which links were possible. The information from the third source, i.e.,

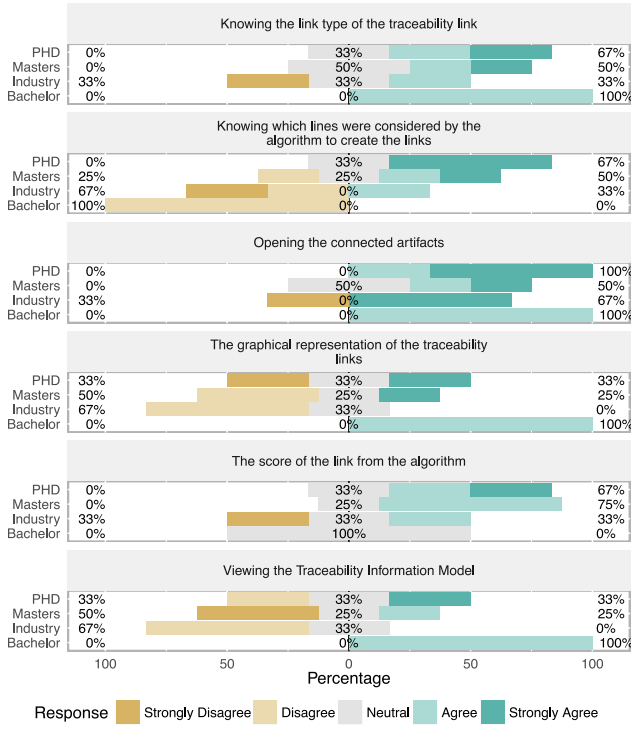

Figure 4. Comparison of Industry participants vs. student participants

the tracing algorithm, such as the similarity score was also perceived as useful by the experiment participants which is in line with the study by Hayes et al. [15].

IX. Threats to Validity

In this section we discuss the threats to validity that are relevant to our study. Since we used multiple research methods we discuss the threats in a combined manner.

Construct Validity: We selected interviewees who had experience with tracing and explained the study prior to the interview. For the experiment, a preliminary session introduced traceability and the aim of the study. However, some participants did not follow the instructions of the experiment. To limit this threat to construct validity, we viewed the screen videos of all participants to exclude these participants.

Internal Validity: The aim of the experiment was to evaluate if the context information makes the analyst more effective. There are four confounding factors that could affect the results of the experiment: tracing experience of the analyst, implementation of link vetting features in Eclipse Capra, the tracing strategy used by the analyst, and the system used for the experiment (MedFleet). While the study by Dekhtyar et al. [20] shows that tracing experience has no effect on the performance of the analyst, it also used students with limited tracing experience. In our study, only eight of the experiment participants had experience with tracing. Conducting the experiment with traceability experts could lead to different results. Regarding Eclipse Capra, we made sure that we only implemented features that were requested by the interviewees and also evaluated existing tools to see how these features were previously implemented. Regarding the tracing strategy, the experiment participants were asked to vet the list of trace links from top to bottom. This ensured that the analysts did not skip any links without investigating them. However, the tracing strategies still varied as some analysts skipped the links they did not understand immediately, while others investigated the link for a longer time before making the decision of skipping the links. This resulted in a variation on the number of links investigated.Regarding the system used for the experiment, we selected a system containing artifacts that are as close to reality as possible. Even though MedFleet was developed by students, it was in a course where students learned software development skills and had to follow proper software development procedures just like in industry. However, this does not guarantee that the system selected had no effect on the study. To properly rule out this internal validity threat, further experiments are needed with different systems.

External Validity: We took several steps to ensure that our study included diverse participants. We interviewed practitioners with different roles, from different companies, and different countries. However, since not all interviewed participants had experience with trace links vetting, we cannot generalize that we elicited all possible context information. Additionally, since the number of interviewees we had was low, we cannot generalize the results. In the experiment, we used participants from different companies and two different universities, with different levels of education and development experience.

Reliability: We documented our process in the research method section (Section III) and have also published our interview guide to make sure that our study can be repeated. Our prototype tool is also accessible as a virtual machine, meaning that the experiment can be repeated by other researchers.

X. Conclusion

In this study, we investigated what information is useful to human analysts when vetting automatically generated traceability links. We specifically investigated if context information can improve the analyst's vetting performance when vetting traceability links. Our interview results reveal three sources of information that can be useful to the analyst. However, our experiment presents evidence to support the conclusion, though not statistically significantly, that context information does not make the analyst more effective during the vetting process. The experiment also shows that analysts made their decisions on trace links by reading the content of the artifacts rather than using context information. Our study shows that the experience of the analyst with the particular system matters more than their general software development experience or tracing experience. We conclude that since the vetting process is a human-centric process, further studies are needed to investigate how this process can be improved by traceability tools to make the analyst more effective.

ACKNOWLEDGEMENTS

Funding for this work has been partially provided by the US National Science Foundation under grants CNS-1649008, CCF-1647342, and CCF-1511117.

REFERENCES

[1] O. C. Z. Gotel and A. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of the First IEEE International Conference on Requirements Engineering, ICRE '94, Colorado Springs, Colorado, USA, April 18-21, 1994*, 1994, pp. 94–101. [Online]. Available: https://doi.org/10.1109/ICRE.1994.292398

[2] B. Ramesh and M. Jarke, "Toward reference models of requirements traceability," *IEEE Trans. Software Eng.*, vol. 27, no. 1, pp. 58–93, 2001. [Online]. Available: https://doi.org/10.1109/32.895989

[3] J. Cleland-Huang, O. C. Gotel, J. Huffman Hayes, P. Mäder, and A. Zisman, "Software traceability: trends and future directions," in *Proceedings of the on Future of Software Engineering*. ACM, 2014, pp. 55–69.

[4] C. Ingram and S. Riddle, "Cost-benefits of traceability," in *Software and Systems Traceability*. Springer, 2012, pp. 23–42.

[5] A. Egyed, S. Biffl, M. Heindl, and P. Grünbacher, "Determining the cost-quality trade-off for automated software traceability," in *Proceedings of the 20th IEEE/ACM International Conference on Automated software engineering (ASE '05)*. ACM, 2005, pp. 360–363.

[6] A. D. Lucia, A. Marcus, R. Oliveto, and D. Poshyvanyk, "Information retrieval methods for automated traceability recovery," in *Software and Systems Traceability.*, 2012, pp. 71–98. [Online]. Available: https://doi.org/10.1007/978-1-4471-2239-5_4

[7] A. D. Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Recovering traceability links in software artifact management systems using information retrieval methods," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 16, no. 4, p. 13, 2007.

[8] M. Borg, P. Runeson, and A. Ardö, "Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability," *Empirical Software Engineering (ESE)*, vol. 19, no. 6, pp. 1565–1616, 2014.

[9] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, "A machine learning approach for tracing regulatory codes to product specific requirements," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering ICSE 2010*, 2010, pp. 155–164.

[10] J. Guo, J. Cheng, and J. Cleland-Huang, "Semantically enhanced software traceability using deep learning techniques," in *Proceedings of the 39th International Conference on Software Engineering, (ICSE '17)*, 2017, pp. 3–14. [Online]. Available: https://doi.org/10.1109/ICSE.2017.9

[11] G. Spanoudakis, A. Zisman, E. Pérez-Miñana, and P. Krause, "Rule-based generation of requirements traceability relations," *Journal of Systems and Software*, vol. 72, no. 2, pp. 105–127, 2004. [Online]. Available: https://doi.org/10.1016/S0164-1212(03)00242-5

[12] R. Michael, R. Jacob, G. Jin L.C., C. Jane, and M. Patrick, "Traceability in the wild: Automatically augmenting incomplete trace links," in *Proceedings of the 40th ACM/IEEE International Conference on Software Engineering (ICSE '18)*, 2018.

[13] A. De Lucia, A. Marcus, R. Oliveto, and D. Poshyvanyk, "Information retrieval methods for automated traceability recovery," in *Software and systems traceability*. Springer, 2012, pp. 71–98.

[14] D. Cuddeback, A. Dekhtyar, and J. Hayes, "Automated requirements traceability: The study of human analysts," in *18th IEEE International Requirements Engineering Conference (RE' 10)*. IEEE, 2010, pp. 231–240.

[15] J. H. Hayes, A. Dekhtyar, S. K. Sundaram, E. A. Holbrook, S. Vadlamudi, and A. April, "Requirements tracing on target (retro): improving software maintenance through traceability recovery," *Innovations in Systems and Software Engineering*, vol. 3, no. 3, pp. 193–202, 2007.

[16] 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 (TEFSE '05)*. ACM, 2005, pp. 20–23.

[17] A. Mahmoud, "Toward an effective automated tracing process," in *20th IEEE International Conference on Program Comprehension (ICPC '12)*, June 2012, pp. 269–272.

[18] N. Niu, W. Wang, and A. Gupta, "Gray links in the use of requirements traceability," in *Proceedings of the 24th International Symposium on Foundations of Software Engineering*, 2016, pp. 384–395.

[19] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *International Symposium on Handheld and Ubiquitous Computing (HUC '99)*. Springer, 1999, pp. 304–307.

[20] A. Dekhtyar, O. Dekhtyar, J. Holden, J. H. Hayes, D. Cuddeback, and W.-K. Kong, "On human analyst performance in assisted requirements tracing: Statistical analysis," in *19th IEEE International Requirements Engineering Conference (RE' 11)*. IEEE, 2011, pp. 111–120.

[21] W.-K. Kong, J. Huffman Hayes, A. Dekhtyar, and J. Holden, "How do we trace requirements: an initial study of analyst behavior in trace validation tasks," in *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '11)*. ACM, 2011, pp. 32–39.

[22] W.-K. Kong, J. H. Hayes, A. Dekhtyar, and O. Dekhtyar, "Process improvement for traceability: A study of human fallibility," in *20th IEEE International Requirements Engineering Conference (RE' 12)*. IEEE, 2012, pp. 31–40.

[23] A. Dekhtyar and M. Hilton, "Human recoverability index: a tracelab experiment," in *Proceedings of the 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE '13)*. IEEE, 2013, pp. 37–43.

[24] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, "Software traceability with topic modeling," in *Proceedings of the 32nd ACM/IEEE international conference on Software Engineering (ICSE '10)*. ACM, 2010, pp. 95–104.

[25] Y. Shin and J. Cleland-Huang, "A comparative evaluation of two user feedback techniques for requirements trace retrieval," in *Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012*, 2012, pp. 1069–1074. [Online]. Available: http://doi.acm.org/10.1145/2245276.2231943

[26] G. Bavota, L. Colangelo, A. De Lucia, S. Fusco, R. Oliveto, and A. Panichella, "Traceme: traceability management in eclipse," in *28th IEEE International Conference on Software Maintenance (ICSM '12)*. IEEE, 2012, pp. 642–645.

[27] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Adams re-trace: A traceability recovery tool," in *9th European Conference on Software Maintenance and Reengineering (CSMR '05)*. IEEE, 2005, pp. 32–41.

[28] J. Lin, C. C. Lin, J. Cleland-Huang, R. Settimi, J. Amaya, G. Bedford, B. Berenbach, O. B. Khadra, C. Duan, and X. Zou, "Poirot: A distributed tool supporting enterprise-wide automated traceability," in *14th IEEE International Requirements Engineering Conference (RE' 06)*. IEEE, 2006, pp. 363–364.

[29] A. Mahmoud and N. Niu, "Tracter: A tool for candidate traceability link clustering," in *19th IEEE International Requirements Engineering Conference (RE' 11)*. IEEE, 2011, pp. 335–336.

[30] N. Niu, A. Mahmoud, Z. Chen, and G. Bradshaw, "Departures from optimality: Understanding human analyst's information foraging in assisted requirements tracing," in *Proceedings of the 2013 International Conference on Software Engineering (ICSE '13)*, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 572–581. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486788.2486864

[31] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[32] R. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014. [Online]. Available: https://doi.org/10.1007/978-3-662-43839-8

[33] S. Maro and J.-P. Steghöfer, "Capra: A configurable and extendable traceability management tool," in *24th IEEE International Requirements Engineering Conference (RE' 16)*. IEEE, 2016, pp. 407–408.

[34] J. Cleland-Huang and M. Rahimi, "A case study: Injecting safety-critical thinking into graduate software engineering projects," in *39th IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training Track, ICSE-SEET 2017, Buenos Aires, Argentina, May 20-28, 2017*, 2017, pp. 67–76. [Online]. Available: https://doi.org/10.1109/ICSE-SEET.2017.4

[35] M. Rahimi, W. Xiong, J. Cleland-Huang, and R. Lutz, "Diagnosing assumption problems in safety-critical products," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE '17)*. IEEE Press, 2017, pp. 473–484.

[36] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1, no. 1.

[37] G. Regan, F. McCaffery, K. McDaid, and D. Flood, "The development and validation of a traceability assessment model," in *Software Process Improvement and Capability Determination*, A. Mitasiunas, T. Rout, R. V. O'Connor, and A. Dorling, Eds. Cham: Springer International Publishing, 2014, pp. 72–83.

[38] M. P. Fay and M. A. Proschan, "Wilcoxon-mann-whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules," *Statistics surveys*, vol. 4, p. 1, 2010.