AUTOMATED REQUIREMENTS TRACEABILITY: THE STUDY OF
HUMAN ANALYSTS

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

David Cuddeback

May 2010
<table>
<thead>
<tr>
<th>COMMITTEE MEMBERSHIP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE:</td>
<td>Automated Requirements Traceability: the Study of Human Analysts</td>
</tr>
<tr>
<td>AUTHOR:</td>
<td>David Cuddeback</td>
</tr>
<tr>
<td>DATE SUBMITTED:</td>
<td>May 2010</td>
</tr>
<tr>
<td>COMMITTEE CHAIR:</td>
<td>Dr. Alex Dekhtyar</td>
</tr>
<tr>
<td>COMMITTEE MEMBER:</td>
<td>Dr. Clark Turner</td>
</tr>
<tr>
<td>COMMITTEE MEMBER:</td>
<td>Dr. Gene Fisher</td>
</tr>
</tbody>
</table>
Abstract

Automated Requirements Traceability: the Study of Human Analysts

by

David Cuddeback

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. Automating the process can save time and potentially improve the quality of the results. When using RTMs to support the V&V of mission- or safety-critical systems, however, a human analyst is required to vet the candidate RTMs. The focus thus becomes the quality of the final RTM. This thesis introduces an experimental framework for studying human interactions with decision support software and reports on the results from a study which applies the framework to investigate how human analysts perform when vetting candidate RTMs generated by automated methods. Specifically, a study was undertaken at two universities and had 33 participants analyze RTMs of varying accuracy for a Java code formatter program. The study found that analyst behavior differs depending on the initial candidate RTM given to the analyst, but that all analysts tend to converge their final RTMs toward a hot spot in the recall-precision space.
Acknowledgements

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.
Contents

List of Tables ix
List of Figures x

1 Introduction 1

2 Background and Related Work 6
  2.1 Measures 7
  2.2 Automated Traceability as Information Retrieval 10
  2.3 The Human Side of Automated Traceability 13
  2.4 Human-Centric Goals of a Tracing Tool 14
    2.4.1 Validation 16
  2.5 Other Traceability Efforts 18

3 Methodology 19
  3.1 Overview 19
  3.2 Research Tool 20
    3.2.1 Modifications 22
  3.3 Dataset 25
  3.4 Candidate RTM Preparation 27
  3.5 Procedure 29
  3.6 Data Collection 30

4 Results 33
  4.1 Overview of Results 33
    4.1.1 Recall-Precision Drift 34
    4.1.2 Improving Recall and Precision 34
List of Tables

2.1 Acceptable levels for recall, precision, and lag. .......................... 8
2.2 Summary of research in IR techniques for automated traceability. 11
2.3 Metrics used to validate goals of a tracing tool. ......................... 18
3.1 Summary of experimental cohorts. ............................................ 20
4.1 Effects on RTM recall and precision by university. ...................... 44
D.1 Frequency of actions taken for true links. ............................... 114
D.2 Frequency of actions taken for false links. .............................. 115
List of Figures

1.1 Traceability identifies links between elements of two artifacts. . . . 2
2.1 The tracing process. ....................................................... 7
2.2 Results from a pilot study. ............................................. 14
3.1 RETRO’s “trace” tab has facilities for automated tracing. ............ 21
3.2 RETRO’s “browse” tab allows for manual tracing. ...................... 22
3.3 Pseudo-code for building an RTM of desired recall and precision from a baseline RTM. ..................................................... 24
3.4 Samples of a requirement and a system test from the experimental dataset. ................................................................. 26
3.5 The pool of candidate RTMs that can be assigned to participants. 28
4.1 Change in recall and precision for all participants by cohort. . . . 45
4.2 Change in recall and precision for participants who increased and decreased the $f_2$-measure. ................................. 46
4.3 Change in recall and precision for participants in the four regions of the recall-precision space. ..................................... 47
4.4 Final candidate RTM distribution. ..................................... 48
4.5 Change in $f_2$-measure and final $f_2$-measure versus the initial $f_2$-measure. ................................................................. 49
4.6 Change in $f_2$-measure versus initial recall and precision. . . . 50
4.7 Final $f_2$-measure versus initial recall and precision. ............... 51
4.8 Final versus initial candidate RTM sizes. ............................ 52
4.9 The change in $f_2$-measure versus the initial candidate RTM size. 53
4.10 Final $f_2$-measure and change in $f_2$-measure versus effort. 54
4.11 Frequency that individual links occur in final RTM versus the initial RTM for true and false links. 55
4.12 Change in recall and precision by university. 56

D.1 Results of all participants. 109
D.2 Analyst effort vs. starting and ending size of the RTM. 110
D.3 Analyst effort vs. size of the initial RTM and final RTM. 111
D.4 Analyst effort vs. the change in size of the RTM. 112
D.5 Analyst effort vs. the recall and precision of the initial RTM. 112
D.6 $f_2$-measure of the final RTM vs. the size of the initial and final RTM. 113
Chapter 1

Introduction

Requirements traceability is “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)” [10]. Requirements traceability is often captured as a requirements traceability matrix (RTM), which specifies the connections between elements of two artifacts in a project, depicted in Figure 1.1. An up-to-date RTM is useful in many contexts, such as verification and validation (V&V), independent verification and validation (IV&V), reverse engineering, criticality analysis, risk assessment, change impact analysis, test coverage analysis, program comprehension, maintenance, and reuse of existing software [1, 12].

In many mission- or safety-critical systems, an up-to-date RTM can be extremely important or even mandated by government regulations. For example, in the 1990s, a telecommunications company in Japan implemented some legally-required changes, which resulted in crashed hardware. The failure was determined to have been caused by developers not being able to trace the changes back to
performance requirements that had to be reevaluated [3].

Unfortunately, generating requirements traceability information post-hoc is an arduous process, because analysts must look at every combination of links between elements. For example, to generate traceability information between 10 requirements and 5 system tests requires an analyst to look at $10 \times 5 = 50$ pairs of elements to decide for each pair whether one satisfies the other. If the number of elements of each document doubles, then the size of the tracing task increases by a factor of four. For large artifacts, the scale of the tracing task can easily be an enormous undertaking.

In routinely executed repetitive tasks, humans have a tendency to make errors [9], which makes repetitive tasks, such as requirements tracing, prime candidates for automation. Automating the process can save time and hopefully improve the quality of the traceability information. To automate the process, researchers have applied techniques from the field of information retrieval. Information retrieval is concerned with which documents from a collection of documents are relevant to a query. In automated requirements traceability, the requirements act as queries and the elements of the other artifact act as the documents to be searched [13].
Research has shown that information retrieval techniques can be effectively applied to generate candidate RTMs in an automated fashion for textual artifacts [1, 20, 12]. 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 [12, 27, 8]. Research that focuses on improving automated traceability methods is often called “the study of methods” [12]. 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.

The research in this thesis addresses the question of whether the analysts will
make the right decisions. To that end, this thesis 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 modified a requirements tracing tool, REquirements TRacing On target (RETRo), to allow 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 33 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 thesis.

The research methodology described in this thesis 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 broad area of software engineering which exhibit these characteristics.

The contributions of this thesis are three-fold: (1) it establishes a framework for studying human interaction with decision support software, (2) it applies the framework to study analyst interactions with automated tracing tools by

\(^{1}\)Our work was approved by the IRB at each University.
\(^{2}\)Understood in a broad sense here.
conducting experiments at two different universities in the United States, and (3) it reports and analyzes the results from those studies.

The rest of the thesis is organized as follows: Chapter 2 covers background information on automated traceability and related research. Chapter 3 describes our research method. Chapter 4 reports the results of our study. Chapter 5 covers the threats to our study’s validity, and finally Chapter 6 contains our concluding remarks and future work.
Gotel defines \textit{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)” [10].

\textit{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 \textit{artifact} can be any by-product of a software life cycle, including a requirements document, design document, and code. An \textit{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 \textit{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
1. Identify the elements of both artifacts to be traced.
2. Assign unique identifiers for each traceable element.
3. For each high-level element, locate all matching low-level elements.
4. For each element in the child document, locate a parent element.
5. Determine if each high-level element has been satisfied.
6. Prepare a traceability matrix.
7. Prepare a summary of the traceability between the two documents.

**Figure 2.1: The tracing process [12].**

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*.

Hayes et al. [12] describe the tracing process as the seven steps in Figure 2.1. Steps 3 and 4 require an analyst to evaluate every combination of links. Steps 3–5 are the focus of research in automated traceability.

### 2.1 Measures

This section mathematically defines the metrics that will be used in this thesis. Many metrics are used to evaluate the quality of an automated tracing tool. The standard metrics of recall and precision are used to measure the tool’s accuracy,
<table>
<thead>
<tr>
<th>Metric</th>
<th>Acceptable</th>
<th>Good</th>
<th>Excellent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>60–69%</td>
<td>70–79%</td>
<td>80–100%</td>
</tr>
<tr>
<td>Precision</td>
<td>20–29%</td>
<td>30–49%</td>
<td>50–100%</td>
</tr>
<tr>
<td>Lag</td>
<td>3–4</td>
<td>2–3</td>
<td>0–2</td>
</tr>
</tbody>
</table>

Table 2.1: Acceptable levels for recall, precision, and lag [12].

of which $f$-measure is a harmonic mean. Additional metrics, selectivity and lag, are used to objectively measure the ease of use and the effort required by the human analyst. Table 2.1 shows the acceptable levels of recall, precision, and lag. These were defined by Hayes et al. [12] based on industry experience by one of the authors. The levels of acceptability have not been empirically determined.

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 $\mathcal{D}$ [12].

**Recall.** Recall is the percentage of correct links that are found [12]. 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 (2.1). Intuitively, recall is the percentage of correct links that were discovered during the tracing process. It is the sum of the correct links found for each requirement divided by the sum of the actual number of correct links for each requirement [12].

$$\text{recall} = \frac{\sum_{q \in \mathcal{H}} r_q}{\sum_{q \in \mathcal{H}} R_q} \quad (2.1)$$
**Precision.** *Precision* is the percentage of retrieved candidate links that are correct [12]. 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 (2.2). Overall precision is the sum of the correct links found divided by the total number of links found [12].

\[
\text{precision} = \frac{\sum_{q \in \mathcal{H}} r_q}{\sum_{q \in \mathcal{H}} n_q} \tag{2.2}
\]

**f-measure.** *f-measure* is the harmonic mean of recall and precision. Defined formally in (2.3), 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.

\[
f_b = \frac{1 + b^2}{\frac{1}{\text{recall}} + \frac{b^2}{\text{precision}}} \tag{2.3}
\]

In this thesis, following Hayes, Dekhtyar, and Sundaram [12], we use the $f_2$-measure, i.e., the *f-measure* for $b = 2$ [12]. $f_2$ is used because better recall means fewer errors of omission while better precision means fewer errors of commission, and we observe that fixing errors of omission is harder than fixing errors of commission.

**Selectivity.** When manually tracing requirements, an analyst must examine all combinatorial links. To trace an artifact with $M$ elements to an artifact with $N$ elements, the analyst must evaluate $M \times N$ candidate links. *Selectivity* is the fraction of those links that are returned by an automated method. Lower values
are better, because they indicate fewer links that have to be examined. Defined formally in (2.4), selectivity is the sum of the number of links returned for each requirement divided by the total number of combinatorial links [12].

$$selectivity = \frac{\sum_{q \in H} n_q}{M \cdot N} \quad (2.4)$$

**Lag.** *Lag* is a measure of the separation between true and false links. For a requirement $q$, $(q, d)$ represents a true link in the list of candidate links. $\text{lag}(q, d)$, the lag of an individual link $(q, d)$, is the number of false links that have higher relevance scores than $(q, d)$. The overall lag for an artifact is the average of the lags for all true links in the list of candidate links. Let $T$ be the set of true links in the candidate links. Equation (2.5) defines the overall lag [12].

$$\text{lag} = \frac{\sum_{(q,d) \in T} \text{lag}(q, d)}{|T|} \quad (2.5)$$

### 2.2 Automated Traceability as Information Retrieval

When textual artifacts are traced to each other, the problem of requirements tracing can be reduced to the problem of information retrieval. Information retrieval is concerned with which documents from a collection of documents are relevant to a query [11]. In requirements tracing, the high-level requirements act as queries, and the low-level elements are the collection of documents [13]. Research has shown that information retrieval techniques are efficient and effective
Table 2.2: Summary of research in IR techniques for automated traceability.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Dataset</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF-IDF [1]</td>
<td>LEDA, Albergate</td>
<td>86–100%</td>
<td>6–18%</td>
</tr>
<tr>
<td>Probabilistic Method [1]</td>
<td>LEDA, Albergate</td>
<td>94–100%</td>
<td>6–19%</td>
</tr>
<tr>
<td>Probabilistic Method [5]</td>
<td>IBS</td>
<td>87%</td>
<td>51%</td>
</tr>
<tr>
<td>TF-IDF with feedback [12]</td>
<td>MODIS [22, 18], CM-1 [21]</td>
<td>90%</td>
<td>80%</td>
</tr>
</tbody>
</table>

Antoniol et al. [1] applied term frequency-inverse document frequency (TF-IDF) [2] and a probabilistic IR method [23] to trace source code to documentation and requirements in two case studies. First, they applied the methods to trace C++ classes to manual pages for LEDA (Library of Efficient Data types and Algorithms). This case study involved 95 KLOC, 208 classes, and 88 manual pages. The second case study was the Albergate system, a hotel management system developed according to the waterfall process and written in Java. It consisted of 20 KLOC and 95 classes, 60 of which were traced to 16 functional requirements. Their results showed high levels of recall (86–100%) but low precision (6–19%) for both methods.

Marcus and Maletic [20] applied another information retrieval technique, latent semantic indexing (LSI) [6], to automate tracing from documentation to source code for the same data sets used by Antoniol et al. [1]. It is worth noting that Marcus and Maletic traced in the opposite direction of Antoniol et al., partially due to limitations of LSI. They also included comments in their research,
whereas Antoniol et al. did not. This may contribute to any differences between their results, especially for LEDA in which the documentation was generated from comments in the source code. In the LEDA case study, LSI achieved comparable recall (96–97%) and higher precision (18–25%) than TF-IDF or the probabilistic model applied by Antoniol et al. In the Albergate case study, recall and precision measures for LSI were comparable to TF-IDF and the probabilistic model (91–100% and 16–17%, respectively).

Cleland-Huang et al. [5] used a probabilistic retrieval algorithm to automate the tracing of non-functional requirements. Non-functional requirements were modeled as a Softgoal Interdependency Graph (SIG) and traced to UML class diagrams for the Ice Breaker System, which comprises 180 SIG elements. After training their methods on a subset of the data, the authors obtained 87% recall and 51% precision on the whole dataset. The authors have used their techniques to study change impact analysis on non-functional requirements [5, 4] and have built a tool to perform automated tracing, called Poirot [19].

Hayes et al. [12] used TF-IDF, TF-IDF with a simple thesaurus, and LSI for requirements to requirements traceability. In addition, they used feedback analysis to incorporate analyst feedback into the results. Relevance feedback analysis is a technique for using the user feedback to improve the accuracy of the information retrieval algorithms. They used the Standard Rochio feedback [2] method for TF-IDF and a variation of it for LSI. The results showed that (a) IR techniques are a practical solution for requirements to requirements traceability and (b) filtering results using user feedback automatically fixed some errors in the original results. In the studies conducted by Hayes et al. [12], TF-IDF vector retrieval resulted in comparable recall and higher precision than LSI. Hayes et al. have built an automated tracing tool called RETRO [16], which the authors use.
for their research.

### 2.3 The Human Side of Automated Traceability

Our research direction, introduced by Hayes et al. [15, 14], 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 [15]. While the study involved only four analysts, it showed that analyst behavior is a problem worthy of further research [15]. Although the sample size was too small to draw conclusions, we show the results from that study in Figure 2.2. 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 vetting. The key observation from the study is that analysts make both errors of omission (throwing out correct links) and errors of commission (adding incorrect links) [15].

On the other side of the interface between the human and the automated tool, Dekhtyar, Hayes, and Larsen [7] 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 trace-
ability data so far is anecdotal. Our work extends the aforementioned study [15] by conducting a more rigorous study involving more participants.

### 2.4 Human-Centric Goals of a Tracing Tool

The human analyst has the final say regarding the traceability of two artifacts and can override any decisions made by an automated tracing tool. He can remove links from the set of candidate links that he believes are incorrect, and he can add links that he believes the tool missed. For better or worse, any decision the analyst makes affects the traceability results.
Recognizing that human analysts affect the final traceability, Hayes et al. [12] identified goals of a tracing tool and measures for validating that a tool satisfies the goals. The philosophy used by Hayes et al. [12] to determine the goals is that a tool should handle the mundane, time-consuming parts of the process so that the human can focus on the parts that require human decision-making.

The goals are discussed below, and their validation measures are discussed in section 2.4.1.\footnote{For the goals of a tracing tool and their validation, see generally to [12].}

**Believability.** A tool shall regenerate candidate links based on analyst feedback such that the final result accurately reflects the theoretical “true trace.” The final results should be trusted. If an analyst doesn’t trust the results of the tool, he will spend more time searching for additional links. Believability consists of three subgoals: accuracy, scalability, and utility.

**Accuracy.** The tool shall return as many true links as possible, while limiting the number of incorrect links that it returns.

**Scalability.** The accuracy and performance of the tool shall not degrade significantly between small and large tracesets. A large traceset is any traceset with more than 3000 combinatorial links. This cutoff is defined by Hayes et al. based on industry experience of one of the authors and on proprietary information [12].\footnote{This goal was included because common criticism of the research in software engineering is that the controlled experiments are too small and “toy-like” [24].}

**Utility.** How much the analyst believes that the tool has helped him achieve accurate traceability. Utility is affected by multiple sub-elements, which are “Op-
erability,” “Process Enforcement,” and “Usefulness.” Operability is the extent to which the user can control and operate the software. Process Enforcement refers to the tool guiding the analyst through the tracing process. Usefulness means that the tool must convince the analyst that it is better to use the tool than to not use it. Hayes et al. haven’t yet defined all the sub-elements of utility.

Discernibility. A tool shall display candidate links and their similarity measures in a way that helps the analyst discern true links from false links. Over time, the true links shall rise to the top and false links shall sink to the bottom of the candidate links. That is, over time, the similarity measures for true links shall be greater than the similarity measures for false links. Lastly, it should be possible to determine the cut-off between true and false links based on their similarity measures. Discernibility prevents the analyst from making incorrect decisions.

Endurability. The tool shall make the tracing process not arduous. Requirements tracing is often neglected because it is arduous, tedious, and boring [14]. If a tool meets all other goals but the task is still pain-staking, then it is likely that tracing will not be performed.

2.4.1 Validation

Hayes et al. [12] also suggest measures to validate that a tool meets the goals of a tracing tool. The goals and their validation metrics are summarized in Table 2.3, and the validation criteria are discussed below.
Believability. Believability consists of accuracy, scalability, and utility. Validating those three sub-goals is sufficient for validating believability. Accuracy is measured by the standard metrics of recall and precision, or alternatively $f$-measure, which represents a balance between the two. For scalability, the tool’s accuracy must not degrade significantly from small to large tracesets. The separation between small and large tracesets is defined to be 3000 combinatorial links. Validating utility consists of objective and subjective measures. Objectively, the tool’s usefulness can be validated by its selectivity. Subjectively, validating utility requires the study of users.

Discernibility. One method for measuring discernibility is to measure the extent to which true links rise to the top and false links sink to the bottom of the candidate link list. This is measured by lag, which is the average number of false links with higher relevance scores than true links in the candidate link list. Lastly, the relevance measures for candidate links should reflect a cut-off between true and false links. This is studied using various techniques to filter candidate links with low relevance scores.

Endurability. Endurability can be measured both objectively and subjectively. An objective measure for endurability is to measure the time it takes an analyst to complete a traceset using a tool, as well as the number of mental comparisons he must make. The number of mental comparisons is measured by the metric of selectivity. Endurability is measured subjectively as how satisfied the analyst is with the tool.
<table>
<thead>
<tr>
<th>Goal</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Believability::Accuracy</td>
<td>Recall, precision, ( f )-measure</td>
</tr>
<tr>
<td>Believability::Scalability</td>
<td>Recall, precision, ( f )-measure</td>
</tr>
<tr>
<td>Believability::Utility</td>
<td>Selectivity</td>
</tr>
<tr>
<td>Discernibility</td>
<td>Lag</td>
</tr>
<tr>
<td>Endurability</td>
<td>Selectivity</td>
</tr>
</tbody>
</table>

Table 2.3: Metrics used to validate goals of a tracing tool [12].

2.5 Other Traceability Efforts

In another effort to reduce the manpower required for requirements tracing, Heindl and Biffl [17] proposed a manual tracing technique called Value-Based Requirements Tracing (VBRT), which requires analysts to prioritize requirements based on value, risk, and effort. Analysts then trace high priority requirements in more detail than low priority requirements. They studied VBRT in a case study involving a preexisting project at Siemens Austria consisting of 46 functional requirements (non-functional requirements were not specified and thus ignored in the case study). The project members prioritized the 46 requirements and then the investigator traced the requirements to code following VBRT. High priority requirements were traced to methods, medium priority requirements to classes, and low priority requirements to packages. They found that VBRT required 35% of the effort required to trace all requirements to methods.
Chapter 3

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 Chapter 1 and are repeated here:

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?

3.1 Overview

Our experiments were conducted in multiple cohorts, summarized in Table 3.1. The first cohort was a pilot study in May 2009. Before the next cohort of experiments, we changed various aspects of our procedure for logistical rea-
Table 3.1: Summary of experimental cohorts.

<table>
<thead>
<tr>
<th>Cohort</th>
<th>Date</th>
<th>Procedure</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>May 2009</td>
<td>Pilot</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>October 2009</td>
<td>Experimental</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>April 2010</td>
<td>Experimental</td>
<td>7</td>
</tr>
</tbody>
</table>

sons and to make it more practical to run our experiments at other universities, including changes to make it easier to pass the institutional review boards (IRB) at other universities. Due to the difference in procedures, we do not describe or report the results of the pilot experiment. What follows is a description of the latter procedure.

3.2 Research Tool

The automated tracing tool used in our study is RETRO (REquirements TRacing On-target) [16]. It uses vector space retrieval methods to suggest candidate RTMs to the analyst and uses the analyst’s corrections to provide feedback to the retrieval algorithm using Standard Rochio Feedback [12]. RETRO evolved from a research toolkit to an industrial tool targeting IV&V analysts and system maintainers, and it has already been used to establish traceability between requirements and design documents [13, 12] as well as bug reports [26].

RETRO consists of three components: the Information Retrieval Toolkit, feedback processing methods, and the GUI front end. The IR Toolkit and feedback processing methods are written in C++ and the GUI is written in Java using the Eclipse SWT GUI framework. The components communicate by modifying files on the hard disk. The IR Toolkit writes its results to an XML file which is read by the GUI and displayed to the user. The GUI receives input from the user.
Figure 3.1: RETRO’s “trace” tab has facilities for automated tracing.

and modifies the XML file, which is then passed to the feedback processor. The feedback processor modifies files on the disk which are used by the IR Toolkit to perform searches [16].

RETRO’s front end has two modes: one helps the analyst perform automated tracing and the other helps the analyst perform manual tracing. The automated tracing mode, shown in Figure 3.1, lets the analyst work with the results of and provide feedback to the automated tracing methods in the IR Toolkit. The analyst can confirm or reject any candidate link suggested by the automated methods. Rejecting a link corrects for errors of commission. The manual tracing mode, shown in Figure 3.2, is equivalent to performing tracing manually without the help of automated methods. Without the manual tracing mode, analysts
Figure 3.2: RETRO’s “browse” tab allows for manual tracing.

would have no way to add links that were missed by the automated methods. Together, RETRO’s two modes of operation allow analysts to correct any type of error made by the automated methods.

### 3.2.1 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 initial candidate RTMs for the same dataset. To do so, we made two modifications to RETRO: (1) we added a login dialog in which participants enter the ID that we assign them, and (2) we replaced the IR Toolkit and feedback processing
methods to deliver RTMs with predetermined recall and precision based on the participant’s ID.

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.

By starting with the baseline RTM, the candidate RTMs suggested by the modified RTM delivery system are representative of candidate RTMs that might be returned by automated methods. Since the information retrieval methods are not used by the modified back-end, the concept of analyst feedback does not apply. The analyst’s corrections are simply incorporated into the candidate RTM without further feedback processing.

The algorithm for building an RTM from the baseline RTM is given in Figure 3.3. It takes as input the target recall and precision as floating point numbers and the set of combinatorial links partitioned into true positives, true negatives, false positives, and false negatives. These partitions are computed by comparing the baseline RTM against the golden standard. The algorithm modifies the partitions to represent the candidate RTM that is delivered to RETRO’s front end. First, the algorithm calculates the number of correct and incorrect links needed by solving Equations (2.1) and (2.2). The lists of true positives and false positives are sorted in reverse order so that removing from the end of the list removes the link with the lowest relevance score. True negatives and false negatives are shuffled so that removing from the end of the list selects a random link. Next,
def build_rtm(desired_recall, desired_precision, true_pos, true_neg, false_pos, false_neg):
    correct_needed = round(desired_recall * len(true_pos))
    total_needed = round(correct_needed / desired_precision)
    incorrect_needed = total_needed - correct_needed

    reverse_sort(true_pos)
    reverse_sort(false_pos)
    shuffle(true_neg)
    shuffle(false_neg)

    while correct_needed < len(true_pos):
        link = pop(true_pos)
        append(true_neg, link)
    while correct_needed > len(true_pos):
        link = pop(true_neg)
        link.relevance = norm(0.35, 0.20)
        append(true_pos, link)

    while incorrect_needed < len(false_pos):
        link = pop(false_pos)
        append(false_neg, link)
    while incorrect_needed > len(false_pos):
        link = pop(false_neg)
        link.relevance = norm(0.35, 0.20)
        append(false_pos, link)

Figure 3.3: Pseudo-code for building an RTM of desired recall and precision from a baseline RTM.
it adjusts the number of true positives and false positives to match the number of correct and incorrect links that are needed. If fewer links are required, the links with the lowest relevance scores are removed. If more links are required, randomly-selected links are added. Any links that are added are given a fake relevance score which is normally-distributed around 0.35 with a standard deviation of 0.20.

### 3.3 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. The training dataset is available as Appendix B.

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. We selected this project 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 sys-
3.0.2

ChangeStyle does not take more than 10 seconds to format 1000 lines of code.

(a) A sample requirement.

TC-7

Purpose:

Verify that ChangeStyle takes 1 seconds or less to format 100 lines of code.

Procedure:

• Format a class that has about than 100 lines of code. Since comments can be formatted as well they are included.

• Follow the steps below:

Test Data:

<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start BlueJ</td>
<td></td>
<td>BlueJ window open</td>
</tr>
<tr>
<td>Open testcase07.java</td>
<td></td>
<td>Class should appear in BlueJ window</td>
</tr>
<tr>
<td>Open Timer</td>
<td></td>
<td>Timer should be at 00:00:00</td>
</tr>
<tr>
<td>Start Timer and Format test-Case07.java</td>
<td></td>
<td>Confirm button should appear</td>
</tr>
<tr>
<td>Stop Timer when confirm button appears</td>
<td></td>
<td>Timer should display less than 00:00:01.00 or 1 second.</td>
</tr>
</tbody>
</table>

(b) A sample system test.

Figure 3.4: Samples of (a) a requirement and (b) a system test from the experimental dataset.
tem 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*. Figure 3.4 shows a sample requirement and system test from the experimental dataset. The experimental dataset is available as Appendix C.

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. We do not make the RTM available at this time, because it is being used in on-going experiments.

### 3.4 Candidate RTM Preparation

In our study, each participant was asked to review a candidate RTM, henceforth referred to as the *initial RTM*, with precision and recall selected from a predefined pool of possibilities. In Figure 3.5, 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
Figure 3.5: The pool of candidate RTMs that can be assigned to participants. Regions are divided by dotted lines.

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 (2.1) and (2.2). 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, we included the points used in the pilot experiment.
3.5 Procedure

The study took place in five 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. All of the experimental materials, including the informed consent form, instructions, and surveys, are available in Appendix A.

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 to assign the 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.

3.6 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. We leave the full analysis of the meta-information for future work. In this thesis, we analyze a subset of the meta-information and the accuracy of the final RTMs compared to the accuracy of the initial RTM. In particular, 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. The full set of features collected are listed below.

**Initial and Final RTM.** We know the initial candidate RTM that we assigned to each participant, and each participant turned in a final RTM. From those, we know the number of correct and incorrect links included in each RTM, and we can calculate the recall, precision, and $f_2$-measure for each RTM. We can also calculate the differences between the initial and final RTM.

**Pre-Experiment Survey.** On the pre-experiment survey, we asked each participant for:

- their major and grade level in school;
- how many software engineering courses they’ve taken;
- how many of those courses covered requirements traceability;
- what industry experience they have;
- if they were exposed to requirements traceability in industry;
- their current comfort level with requirements traceability;
- when and where they were first exposed to traceability;
- the number of times they’ve performed requirements traceability;
- what methods they used for traceability (hard copy, soft copy, word processor, spreadsheet, CASE tool, and other); and
• any other comments (open-ended).

Post-Experiment Survey. On the post-experiment survey, we asked each participant for:

• the time they spent learning RETRO;

• how often they used the following features of RETRO: trace all, trace current, filtering, sorting, feedback, view documents, and add link;

• opinions on whether: the training exercise prepared them, RETRO reduces effort or time required to trace requirements, tracing requirements with RETRO is easy, their classes prepared them, their industry experience prepared them, and they felt prepared;

• their current comfort level with requirements traceability;

• their familiarity with the ChangeStyle project;

• their input on how RETRO can be improved (open-ended); and

• any other comments (open-ended).

Time Sheet. On the time sheet, we asked each participant to tally how many times they started over; to keep track of how much time they spent on preparation, tracing, and wrap-up; and to note any issues they encounter.
Chapter 4

Results

This section presents the results and analysis. Further analysis of the results is left for future work.

4.1 Overview of Results

As shown in Figure 3.5, we collected 33 responses to our study from two cohorts of experiments. The first cohort comprised 26 responses in 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 second cohort involved only one class, from which we received 7 responses. The filled dots in Figure 3.5 represent assigned candidate RTMs from the first cohort, while the stars represent assigned candidate RTMs from the second cohort, and hollow dots represent candidate RTMs that were not assigned. The first and second cohorts are named “cohort 2” and “cohort 3” respectively to be consistent with the numbering used in Table 3.1 (“cohort 1” was the pilot study, on which
we do not report in this thesis). Figures 4.1–4.12 depict the key results of our study. The figures begin on page 45.

4.1.1 Recall-Precision Drift

Figure 4.1 presents an overview of our results, divided by cohort. Each vector plotted on the graphs 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. Figure 4.3 breaks Figure 4.1 by region (quadrant) of the starting candidate RTM where the regions are divided by 63% recall and 63% precision (the same divisions used between the regions in Figure 3.5). Figure 4.2 breaks the results by those who improved the $f_2$-measure and those who did not. Figure 4.4 shows the distribution of the recall and precision of all submitted candidate RTMs.

4.1.2 Improving Recall and Precision

In our study, 14 participants improved the recall of their candidate RTM, four kept it the same, and 15 lowered it. 17 participants improved the precision of their candidate RTM, two participants kept the same precision, and 14 decreased it. Only seven participants improved both recall and precision, while five participants decreased both and one participant kept recall and precision the same.
4.1.3 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). Figure 4.2 breaks up the results by participants who improved the $f_2$-measure and those who reduced it. Overall, 16 participants improved $f_2$ and 17 did not.

Figure 4.5(a) plots the difference in $f_2$ values between the starting and the ending candidate RTMs (denoted as $\Delta f_2$) 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 $f_2$. Figure 4.5(b) plots the final values of $f_2$ vs. the starting values of $f_2$.

4.1.4 Changes by Region

Figure 4.3 shows the changes in recall and precision divided by regions of the recall-precision space. Figures 4.6 and 4.7 show 3D plots and heat maps for $\Delta f_2$ and final $f_2$, respectively. The plots are in the recall-precision space.

4.1.5 RTM Size

Figure 4.8 shows the scatter plots comparing the size of the final candidate RTM to the size of the initial candidate RTM. Figure 4.9 shows the change in the $f_2$-measure based on the size of the initial candidate RTM.
4.1.6 Effort

Figure 4.10 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 as self-reported by the participants. Figure 4.10(b) plots the change in the $f_2$-measure vs. the effort.

4.2 Analysis

Based on the results described briefly in Section 4.1, we make the following observations concerning the results of our study.

4.2.1 Movement Toward $\text{recall} = \text{precision}$ Line

One of the surprising observations made in the prior study [15] was the fact that human analysts tended to move their candidate RTMs toward the $\text{recall} = \text{precision}$ line. With only four data points described [15], this observation required more significant confirmation. In our current study, we clearly observed the same drift.

Why do so many final RTMs seem to hover around the $\text{recall} = \text{precision}$ line? One observation we make is that candidate RTMs with $\text{recall} \approx \text{precision}$ will be about the same size as the true RTM. This is easily proven: setting equations (2.1) and (2.2) 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 $\text{recall} = \text{precision}$.

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 Figure 4.8, 21 out of 33 participants (64%) submitted RTMs containing between 17 and 32 links, some of whom started with a candidate RTM that was either above or below this range.

4.2.2 Hot Spot for Final RTMs

Figure 4.4 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 $f_2$-measure is between 0.60 and 0.75 and the size of the RTM is between 17 and 32.

19 participants submitted RTMs with an $f_2$-measure between 0.60 and 0.75. As mentioned in Section 4.2.1, 21 participants submitted RTMs containing between 17 and 32 links. 15 participants (45%) submitted RTMs contained between 17 and 32 links and had an $f_2$-measure between 0.60 and 0.75.

Final RTMs with $f_2$-measure both higher and lower than the initial RTM’s $f_2$-measure are present in this region (10 and five respectively; with one and two, respectively, just outside this region). Note that only two participants were assigned initial candidate RTMs in this region. Six participants were assigned RTMs with $f_2$-measure between 0.60 and 0.75, and 14 were assigned RTMs containing between 17 and 32 links.
4.2.3 Regional Behavior Differs

We observe distinctly different results for participants who started with candidate RTMs in different regions.

**Low Recall, High Precision.** Participants given initial candidate RTMs from the low recall, high precision region (Figure 4.3(a)) 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. RTMs in this region (low recall, high precision) tend to have very few links, so we conjecture that the participants working with these candidate RTMs spent most of their time looking for errors of omission and consequently introduced both true links and false positives into their candidate RTMs.

**High Recall, High Precision.** Participants working with high recall, high precision candidate RTMs (Figure 4.3(b)) 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 $f_2$-measure was in the range of 0.45 to 0.9 for all starting RTMs (Figure 4.7), with much of the recall-precision space being colored dark orange (0.65) and hotter.

**Low Recall, Low Precision.** With one exception, participants working with candidate RTMs from the low recall, low precision region (Figure 4.3(c)) drastically improved both precision and recall, and in general, demonstrated the highest
improvement in the accuracy of the final candidate RTM\textsuperscript{1}. This is also seen in Figure 4.6 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. The participant who did not improve both recall and precision significantly improved precision with a minor decrease in recall.

**High Recall, Low Precision.** Finally, participants given initial candidate RTMs from the high recall, low precision region (Figure 4.3(d)) improved, sometimes significantly, the precision of the submitted RTM, but generally decreased recall. One participant significantly reduced recall (from 91% to 9%) and decreased precision. This participant appears to us to be an outlier. Because high recall, low precision 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). In the process of weeding out false positives, some true links were removed from the RTM.

4.2.4 Those Who Improved Accuracy

There were 16 participants who improved the overall accuracy (measured as $f_2$) of their candidate RTM. As Figure 4.5 shows, all but three improvement led to candidate RTMs with an $f_2$-measure value between 0.6 and 0.75. Figure 4.5(a) 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.

\textsuperscript{1}Although, they started with initial candidate RTMs having the lowest accuracy. Thus, they had the most room for improvement.
4.2.5 Those Who Did Not Improve Accuracy

On the other hand, the results of the 17 participants who did not improve the RTM accuracy show a distinctly different pattern. All but three participants showed only a slight decrease in the accuracy of their final candidate RTM (a decrease of 0 to 10%), and this decrease did not depend on the accuracy ($f_2$-measure) of the initial candidate RTM.

4.2.6 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 Figure 4.10(a), most participants (19 out of 33) submitted an RTM with a final $f_2$-measure between 0.6 and 0.75. The effort applied by the participants varied greatly from 15 to 95 minutes.

The participant who expanded the most effort returned an RTM with $f_2$-measure of only 0.25, while the participant who applied the least effort returned an RTM with a final $f_2$ of 0.6. From Figure 4.10(a), we observe a cluster of participants who spent anywhere from 35 to 80 minutes on the task and achieved final $f_2$-measure of 0.6 to 0.82.

Change in $f_2$-measure is even more telling (see Figure 4.10(b)). Participants who applied 30 to 70 minutes of effort yielded -0.1 to +0.4 change to $f_2$ (the two participants with -0.5 $\Delta f_2$ are believed to be outliers). We conclude, therefore, that there is no visible correlation between the effort applied and the final $f_2$ or the change in $f_2$. 

40
4.2.7 Link Quality

In Section 4.2.2, we identified that 19 out of the 33 participants produced final RTMs with \( f_2 \)-measure between 0.60 and 0.75. Only four participants turned in final RTMs with \( f_2 \)-measure better than 0.75. Figure 4.5(b) makes it clear that the only participants who produced RTMs better than 0.75 \( f_2 \)-measure started with initial candidate RTMs that were better than 0.75 \( f_2 \)-measure.

We hypothesize that the value of this ceiling (0.75 \( f_2 \)-measure in the case of these experiments) is due to a property of the dataset itself. In particular, we hypothesize that it is determined by the quality of the links in the dataset. That is, some links may be harder to detect than others.

Figure 4.11(a) plots the number of final RTMs containing each true link versus the number of initial RTMs containing the same link. Table D.1 shows the same information in tabular form as well as the number of people who added, removed, kept, or missed each link. The general trend in Figure 4.11(a) shows a correlation between a true link being in the final RTM and starting with it in the initial RTM.

There are a few notable outliers. Link #20 started in 15 initial candidate RTMs, but was discarded by 10 participants (67%) and added by none. Links #8 and #17 were both added by 11 people, which are 73% and 42% respectively of participants who did not have those links in their initial RTM. Lastly, link #23 was not in any participant’s initial or final RTMs. The outliers suggest that some true links are more easily detected than others, while some appear outright wrong. These ambiguities will affect the accuracy of the final RTM.

A different pattern emerges for false links, which are shown in Figure 4.11(b) as the number of final RTMs containing each false link versus the number of initial candidate RTMs containing the same link and in tabular form in Table D.2. While
the same overall correlation exists with false links being more likely to appear in the final RTM if they exist in the initial RTM, participants had more of a tendency to remove false links than they did with true links. Some links were removed by more participants than others, suggesting that some links are more clearly wrong than others.

There are also a number of outliers in the false links—some that were added by participants. In particular, links #28, #29, and #40 were added by 15, 12, and 7 participants, respectively. Many links were added by five or fewer participants. Perhaps the most ambiguous links are those that were both added and removed by many participants, such as link #30, which was added and removed by six participants.

These results suggest a number of things. The overall pattern in both true and false links shows that links that begin in the initial RTM tend to remain in the final RTM. The differences between how participants handle true and false links shows that participants are more likely to remove false links than to find missing true links. It also shows that some links are more easily detected as true or false links by the participants—that is, that some links are more ambiguous than others. Lastly, it could indicate an incorrect classification of some of the links in what we consider to be the golden standard RTM (see Chapter 5, Threats to Validity). We plan to investigate this in the future.

4.2.8 Cal Poly Participants vs. Kentucky Participants

The study was conducted at two universities. Figure 4.12 divides the results by university. Overall, we notice that students from the University A 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 outliers in our study: two participants from University A drastically decreased the accuracy of the returned RTM, while a participant from University B submitted the starting RTM without performing any tracing.

The only interesting observed discrepancy occurred for those who received initial candidate RTMs with high recall and high precision (both recall and precision over 75%). The three participants from University B showed a preference for keeping precision (almost) the same, while decreasing, sometimes significantly, the recall. Two of the three 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 eliminating 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).

As can be seen in Figure 4.12, University A had mixed results. Three participants decreased both recall and precision, three participants increased recall and precision, one participant increased precision and did not change recall, five participants increased precision but decreased recall, and two participants made no real change to the precision and recall of the RTM.

Table 4.1 summarizes how the two universities affected recall and precision of the candidate RTMs. There is not a significant difference between the two un-
<table>
<thead>
<tr>
<th>Result</th>
<th>University A</th>
<th>University B</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increased Recall</td>
<td>7 (41.2%)</td>
<td>7 (43.8%)</td>
<td>14 (42.4%)</td>
</tr>
<tr>
<td>Decreased Recall</td>
<td>8 (47.1%)</td>
<td>7 (43.8%)</td>
<td>15 (45.5%)</td>
</tr>
<tr>
<td>Increased Precision</td>
<td>9 (52.9%)</td>
<td>8 (50.0%)</td>
<td>17 (51.5%)</td>
</tr>
<tr>
<td>Decreased Precision</td>
<td>8 (47.1%)</td>
<td>6 (37.5%)</td>
<td>14 (42.4%)</td>
</tr>
</tbody>
</table>

Table 4.1: Effects on RTM recall and precision by university.

versities in most of their effects, with the exception of participants who decreased precision. 47.1% of participants from University A decreased the precision of the candidate RTM that was given to them, while only 37.5% of participants from University B did the same. Although the distribution differs somewhat from that of the overall study population, the variance is not so large to indicate an effect based on university.

### 4.2.9 Outliers Happen Anywhere

We observed that outliers can happen anywhere. Two participants in our studies failed miserably, but failures appear to have no correlation to the accuracy of the initial candidate RTM. Each outlier started with initial candidate RTMs in different regions of the recall-precision space: one outlier received an initial candidate RTM with moderate recall and high precision; the other received one having high recall and low precision. Both outliers finished with RTMs having very low recall and low precision.

The patterns in the regions surrounding the outliers are strong enough to indicate that these are indeed outliers. For example, in the region surrounding the outlier with high recall and low precision, five other participants showed a tendency to increase precision. We plan to study other features in the data that we collected to look for explanations.
Figure 4.1: Change in recall and precision for all participants by cohort.
Figure 4.2: Change in recall and precision for participants who (a) increased and (b) decreased the $f_2$-measure.
Figure 4.3: Change in recall and precision for participants in the four regions of the recall-precision space.
University A  
RTM size = 17  
f₂ = 0.60

University B  
RTM size = 32  
f₂ = 0.75

Figure 4.4: Final candidate RTM distribution.
Figure 4.5: (a) Change in $f_2$-measure and (b) final $f_2$-measure versus the initial $f_2$-measure.
Figure 4.6: Change in $f_2$-measure versus initial recall and precision.
Figure 4.7: Final $f_2$-measure versus initial recall and precision.
Figure 4.8: Final versus initial candidate RTM sizes.
Figure 4.9: The change in $f_2$-measure versus the initial candidate RTM size.
Figure 4.10: (a) Final $f_2$-measure and (b) change in $f_2$-measure versus effort.
Figure 4.11: Frequency that individual links occur in final RTM versus the initial RTM for (a) true and (b) false links.
Figure 4.12: Change in recall and precision for (a) University A and (b) University B.
Chapter 5

Threats to Validity

**Internal Validity.** Influences that may impact the independent variables with respect to causality are referred to as threats to internal validity [25]. A possible personal bias threat in preparation of the golden standard RTM was reduced by having multiple researchers review the answer set. It’s possible that some of the links were miscategorized in the golden standard RTM, as discussed in Section 4.2.6. Personal bias in conducting the study was reduced by using a random assignment of students to RTMs.

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.

The tool itself is a threat to internal validity, because its quality could affect the participants’ abilities to perform the task. Specifically, many participants complained that the way that RETRO displays the test cases makes them difficult
to read. Some participants also reported bugs related to finding new links on the “browse” tab and RETRO crashed for some participants. Although many bugs were fixed between the pilot study and the latter cohorts, it’s still possible that the quality of RETRO could affect the results. We would like to repeat these experiments with a similar tool such as Poirot [19] to corroborate our findings.

**Construct Validity.** 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 Validity.** 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.** 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.
Chapter 6

Conclusions and Future Work

This thesis 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 [15, 14]: 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.

Our second conjecture is that the accuracy of the final RTM depends on clarity of the elements in the artifacts being traced. Indeed, we saw that some links were more consistently classified than others.

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). We also plan to investigate the possibility that our golden standard RTM is incorrect. We’ll look specifically at the links that exhibited the most suspect behavior.

The overarching goal of our study is to determine which factors influence the work of a software engineer with automated tracing tools. In this thesis, 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.

We also need more data. We received few responses from some of the regions in the recall-precision space. In particular, the low recall, medium precision region is represented by only one participant. We also need to collect more data to make data mining a viable method for uncovering patterns in analyst behavior. We plan to conduct further experiments with more participants, focusing on filling
in the gaps in our data. Lastly, we’d like to repeat these experiments with a different tracing tool to eliminate the tool as a variable.
Bibliography


Appendix A

Experimental Materials

A.1 Informed Consent Form

Informed Consent to Participate in a Research Study

*Automated Requirements Traceability: The Study of Human Analysts*

**WHY ARE YOU BEING INVITED TO TAKE PART IN THIS RESEARCH?**

You are being invited to take part in a research study about trace sets. You are being invited to participate in this research study because you learned about requirements development in software engineering courses at Cal Poly. It is your choice whether or not you volunteer for this study. If you choose not to participate, there will not be any ramifications and you will have the opportunity to perform a different extra credit project of equal value and equal difficulty. If you take part in this study, you will be one of about 15 people to do so.
WHO IS DOING THE STUDY?

The people in charge of this study are Dr. Alex Dekhtyar, Professor of Computer Science at California Polytechnic State University in San Luis Obispo, California, and David Cuddeback, graduate student of Computer Science at California Polytechnic State University in San Luis Obispo, California. There may be other people on the research team assisting at different times during the study.

WHAT IS THE PURPOSE OF THIS STUDY?

By doing this study, we hope to obtain information about analyst decision making when working with computer-generated trace sets.

WHERE IS THE STUDY GOING TO TAKE PLACE AND HOW LONG WILL IT LAST?

The research procedures will be conducted during a class lab period. The total amount of time you will be asked to volunteer for this study is approximately 2 hours over the 1 week.

WHAT WILL YOU BE ASKED TO DO?

We are asking you to perform a tracing task. Specific instructions are attached to this form.
ARE THERE REASONS WHY YOU SHOULD NOT TAKE PART IN THIS STUDY?

We will protect your identity. Only Dr. Dekhtyar will know what analyst worked on a specific trace set. We are not aware of any dangers or other risks of which we need to make you aware.

WILL YOU BENEFIT FROM TAKING PART IN THIS STUDY?

As a participant in this study you will receive 1% extra credit in your Software Engineering course. You will not receive any payment for taking part in this study.

WHAT WILL IT COST YOU TO PARTICIPATE?

There are no costs associated with taking part in this study, except your investments of time and intellect.

DO YOU HAVE TO TAKE PART IN THE STUDY?

If you decide to take part in the study, it should be because you really want to volunteer. You will not lose any benefits or rights you would normally have if you choose not to volunteer. If you choose not to volunteer, nobody will be notified or made aware of your choice. Only the researchers will be aware of your choice, and they will keep this information confidential. You can stop at any time during the study and still keep the benefits and rights you had before volunteering.
IF YOU DON'T WANT TO TAKE PART IN THE STUDY, ARE THERE OTHER CHOICES?

If you do not want to be in the study, you may choose to undertake a different extra credit opportunity of equal value and equal difficulty, or you may choose not to undertake any extra credit projects.

WHO WILL SEE THE INFORMATION YOU GIVE?

When we write up the study to share with other researchers, you will not be identified in these written materials. However, there are some circumstances in which we may have to show your information to other people. For example, we may be required to show information which identifies you to people who need to be sure we have done the research correctly; these would be people from such organizations as Cal Poly State University.

We will make every effort to prevent anyone who is not on the research team from knowing that you gave us information, or what that information is. For example, your name will be kept separate from the information you give, and these two things will be stored in different places under lock and key.

CAN YOUR TAKING PART IN THE STUDY END EARLY?

If you decide to take part in the study, you still have the right to decide at any time that you no longer want to continue. You will not be treated differently if you decide to stop taking part in the study.
WHAT IF YOU HAVE QUESTIONS?

Before you decide whether to accept this invitation to take part in the study, please ask any questions that might come to mind now. Later, if you have questions about the study, you can contact the investigator, Dr. Alex Dekhtyar, Professor in the Computer Science Department of Cal Poly State University at 805-756-2387. If you have any questions about your rights as a research volunteer, contact Steve Davis of the Human Subjects Committee at Cal Poly University at sdavis@calpoly.edu, 756-2754, or Dr. Susan Opava, Dean of Research and Graduate Programs at 756-2754, sopava@calpoly.edu. The Human Subjects Committee home page is http://www.calpoly.edu/~sdavis/human.htm. We will give you a copy of this consent form.

<table>
<thead>
<tr>
<th>Signature of person agreeing to take part in the study</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Printed name of person agreeing to take part in the study</td>
<td>Date</td>
</tr>
<tr>
<td>Name of person providing information to subject</td>
<td>Date</td>
</tr>
</tbody>
</table>
A.2 Pre-Experiment Survey

Pre-Study Questionnaire

Automated Requirements Traceability: The Study of Human Analysts

0. What is your name? ____________________________

1. What is your major? (circle one)
   (A) Computer Science          (C) Computer Engineering
   (B) Software Engineering       (D) Other: __________

2. What is your grade level? (circle one)
   Freshman                     Junior         Graduate
   Sophomore                    Senior

3. Which Software Engineering courses have you taken or are currently
   enrolled in? (circle all that apply or “none”)
   CSC-305  CSC-402  CSC-509
   CSC-307  CSC-405  NONE
   CSC-308  CSC-406
   CSC-309  CSC-508

4. In which Software Engineering courses did you build a requirements
   traceability matrix (RTM)? (circle all that apply or “none”)
   CSC-305  CSC-402  CSC-509
   CSC-307  CSC-405  NONE
   CSC-308  CSC-406
   CSC-309  CSC-508

5. What kind of industry experience do you have? (circle all that apply)
(A) Worked professionally before enrolling at Cal Poly.

(B) Work professionally while enrolled at Cal Poly.

(C) Internship/Co-op

(D) None

6. Did you encounter traceability in your industry experience?
   Yes                      No

7. How comfortable do you feel about your ability to perform requirements tracing? (circle one)
   Very uncomfortable            Neither comfortable or uncomfortable
   1                           2                           3
   Very comfortable
   4                           5

8. Where did you encounter requirements traceability for the first time?
   (A) Have not encountered before
   (B) Encountered in a CSC/SE college course (Cal Poly or elsewhere)
   (C) Encountered while on an industry internship or co-op
   (D) Encountered while employed in the industry (full-time or part-time)

9. When did you encounter requirements traceability for the first time?
   (A) Have not encountered before
   (B) Between Fall 2008 and Spring 2009
   (C) Some time between Fall 2005 and Summer 2008
   (D) Some time earlier than Fall 2005
10. How many times have you performed requirements tracing since then?

   (A) Never
   (B) Once
   (C) Two to four times
   (D) Five or more times

11. What did you use when performing requirements tracing? (circle all that apply)

   (A) Did not perform tracing tasks
   (B) Used hard copies of artifacts to perform tracing
   (C) Used soft copies of artifacts to perform tracing
   (D) Used word processor to perform tracing
   (E) Used a spreadsheet or a database to maintain RTM
   (F) Used a CASE tool to assist in tracing
   (G) Used some other tool/technique/method to perform tracing:

      Specify: ________________________________

12. What else should we know about your experience with traceability?
A.3 Post-Experiment Survey

Post-Study Questionnaire

*Automated Requirements Traceability: The Study of Human Analysts*

1. What is your user ID: __________________________

2. Between the completion of the in-lab training session and the beginning of the experimental tracing task, did you spend any time using RETRO?
   
   (A) Yes
   
   (B) No

3. If yes to question 2, how much time did you spend using RETRO?
   
   (A) Less than 30 minutes
   
   (B) 30 minutes to an hour
   
   (C) 1 hour to 2 hours
   
   (D) More than 2 hours
4. Specify how often you used the following RETRO functionality when completing your task.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Did Not Use At All</th>
<th>Used Rarely</th>
<th>Used a Few Times</th>
<th>Used Frequently</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) “Trace All” Button</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(B) “Trace Current” Button</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(C) Filtering of Low-Level Elements</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(D) View candidate links “By Weight”, “In Document Order”</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(E) Specify “Link”, “Not A Link”, “Default” status of candidate links</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(F) Use Browse Tab to view documents</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(G) Use “Add Link” functionality in Browse Tab</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
5. Rate how strongly you agree with the following statements.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly Agree</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) The training session provided me with sufficient information to</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>complete my task.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(B) RETRO reduces the effort required to trace requirements.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(C) RETRO reduces the time required to trace requirements.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(D) Tracing requirements with RETRO is easy.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(E) My classes prepared me to perform the task that I was asked to</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>perform.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) My professional experience prepared me to perform the task that I</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>was asked to perform.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(G) I was prepared to perform the task that I was asked to perform.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. How comfortable do you feel about your ability to perform requirements tracing? (circle one)

<table>
<thead>
<tr>
<th>Very uncomfortable</th>
<th>Neither comfortable or uncomfortable</th>
<th>Very comfortable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

77
7. The CheckStyle program was a recent CSC-308/309 project. Were you familiar with this program before participating in this experiment? (circle one)

(A) Yes, I took CSC-308/309 when this project was assigned.
(B) Yes, I know someone who worked on this project.
(C) No

8. In what ways do you think RETRO can be improved?

---

9. Is there anything else you would like to tell us regarding your experience using RETRO to perform requirements tracing? (This is your chance to expand on any of your answers above.)
A.4 Time Sheet

Time Sheet

Automated Requirements Traceability: The Study of Human Analysts

0. User ID: ______________________

1. Please keep a tally of how many projects you create: _________

2. Time Sheet:

<table>
<thead>
<tr>
<th>Task</th>
<th>Time Started</th>
<th>Time Ended</th>
<th>Time Elapsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tracing</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total:

3. Please record any issues you discover below:

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

79
Retrospectives

Your Instructions

Fall 2009 Automated Traceability Experiment

Instructions for working with RETRO

RETRO

RETRO, a.k.a., Requirements Tracing On Target is a software tool developed at the University of Kentucky for the purpose of supporting the requirements tracing process. RETRO automates candidate link generation on tracing tasks.

Running RETRO

RETRO will run on any Windows PC with the Java JRE installed. To run it on your local machine, perform the following tasks


2. Download RETRO.zip.

3. Extract the downloaded zip file. This should create a folder called RETRO in the location that you choose to extract the zip file.

   (Your desktop is a good choice, as you will only need the files while running the experiment.)

4. Open Windows Explorer and go to the extracted RETRO folder.
5. There is a `RETRO.bat` file in the RETRO folder.

   To run RETRO, double-click on the `RETRO.bat` file icon.

Working With RETRO

Login.  Login as test with password test when getting acquainted with RETRO. For your assignment, you will be issued an account and asked to login under the account name issued to you.

Projects. A single tracing task in RETRO is called a project. To start a new tracing task, you create a new project; to complete an unfinished task you Load a project.

Each project has two artifacts associated with it it: a low-level artifact and a high-level artifact. Tracing takes place from the high-level artifact to the low-level artifact.

Artifacts. RETRO accepts artifacts as a directory of text files. Each file represents a single element (e.g., requirements, system test, bug report, etc). The file name is used as the unique ID of the element in the artifact.

You will be working with the artifacts in the “intro” dataset, available on the web site. To get the artifacts, perform the following tasks.


2. Download `intro-dataset.zip`.

3. Extract the downloaded zip file.
The extracted dataset contains two subdirectories, `low`, storing the low-level artifact (a collection of system tests), and `high`, storing the high-level artifact (a collection of functional requirements).

**Starting a new project.** Upon logging in, RETRO will provide a menu of choices for you.

1. To start a new project, select "Create New Project" button. Then proceed as follows:

2. In the dialog window that opens, enter the following information:

   (a) **Name of the project.** Project names can be any identifier. You can name the project whatever you want, but it can’t be a name you used for a previous project.

   (b) **Low-Level artifact directory.** Select the `low` subdirectory of the intro dataset.

   (c) **High-Level artifact directory.** Select the `high` subdirectory of the intro dataset.

**Trace Tab.** The main screen of RETRO has two tabs **TRACE** and **BROWSE**.

In order to use the **TRACE** tab functionality, you need to perform the automated tracing. Click on the "Trace All" button to start the trace.

**Browse Tab.** The **BROWSE** tab functionality can be used at any time. You can browse the documents and manually link individual high- and low-level elements.
A.6 Experiment Instructions

Traceability Study Instructions

Thank you for participating in the traceability study. The goal of this study is to assess how human analysts interact with special-purpose tracing software. This document outlines your task for the experiment.

Setup

You will be working with the version of RETRO (Requirements Tracing On-Target) tool available from the website

http://users.csc.calpoly.edu/~dcuddeba/RETRO

Download RETRO.zip and extract the downloaded zip file. This should create a folder called RETRO in the location that you choose to extract the zip file. Inside the RETRO folder is the RETRO.bat batch file. You can run RETRO by double-clicking RETRO.bat.

To perform the assigned task you will need to start RETRO and log in using the account ID issued to you. Your account is:

Affix user ID and password here.
Individual Work

Use the lab period to work on the task. If you need time beyond the lab period to complete the task, please complete the task later.

The experiment assesses the work of individual analysts with the software. Therefore, it is crucial to the experimental design that while performing the task outlined below you do not interact with other people participating in the experiment.

If you have specific questions about the nature of the task, or the software, please feel free to approach any member of the research team administering the experiment. We will try to address all your questions.

The Task

You will be using RETRO to trace a set of functional requirements for a Java style-checker plugin for BlueJ to a set of system tests. In this assignment, the functional requirements play the role of the high-level artifact while the system tests play the role of the low-level artifact.

The requirements and system tests can be downloaded from the following URL:

http://users.csc.calpoly.edu/~dcuddeba/RETRO/CheckStyle-dataset.zip

After extracting the zip file, the functional requirements will be in the high subdirectory, and the system tests will be in the low subdirectory. Both directories contain the input data in the format ready for the use with RETRO.
Note: In the dataset you are working with not all functional requirements can be traced to system tests.

Starting New Project

Upon RETRO startup create a new project. Give your project the same name as the RETRO loginID assigned to you. (For example, if your loginID is user22, name your project user22).

Specify the directories for high-level and low-level artifacts as above.

When the main screen of RETRO loads, press “Trace All” button. RETRO will perform an automated tracing task and will load the candidate Requirements Traceability Matrix (RTM) for you.

Tracing

Using the functionality provided to you by RETRO, complete the tracing task.

You can use any tools available to you through RETRO to determine the final (in your opinion) RTM.

Note: If at any point you decide that you need to restart your project from scratch, feel free to do so. You can exit RETRO and restart it (or just open a new project). When opening a new project, please append a “dash” and the ordinal number to the end of your project name. For example, user user22 would start with project user22, and then proceed to user22-1, user22-2, etc., for as long as needed.
Completing the Tracing Task

Once you believe that you have correctly identified the Requirements Traceability Matrix for your task, click the “Complete Trace” button (which will open a tab in which you can review the entire RTM).

In the RTM, each link will be marked as either Link, Not a Link or Default. When analyzing your final RTM, we will make the following determinations:

<table>
<thead>
<tr>
<th>Link Status</th>
<th>Our interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>The link IS part of the final RTM</td>
</tr>
<tr>
<td>Not a Link</td>
<td>The link IS NOT part of the final RTM</td>
</tr>
<tr>
<td>Default</td>
<td>The link IS part of the final RTM</td>
</tr>
</tbody>
</table>

Submitting Your Results

You must submit your results to the research team in order for your results to be counted in the experiment. To submit your results,

1. Click on the “Complete Trace” button. A “View Mode” tab should appear which shows the final RTM.

2. At the bottom of the “View Mode” tab, click the “Submit Results” button. This will ask you to save the RTM to a file.

3. Save the RTM to a file with the same name as your user ID. For example, if your user ID is user22, save the file as user22.xml. This is the default behavior of the save dialog.

4. Email the file you just saved to the research team at the following email address:

   dcuddeba@calpoly.edu

1“Default” state means “Computer thinks it’s a link.”
After we receive your email, we will send you a link to the post-study survey by replying to your email.

**Time Card**

Together with the assignment, you are given a simple time card. We ask you to keep track of your time spent on this assignment and to record it on the time card. We ask you to include the times you spent on the following tasks:

<table>
<thead>
<tr>
<th>Task</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>The time between starting the experiment and starting the actual tracing</td>
</tr>
<tr>
<td>Tracing</td>
<td>Time spent performing tracing.</td>
</tr>
<tr>
<td>Wrap-up</td>
<td>Time spent on wrapping-up your work.</td>
</tr>
</tbody>
</table>

If you created more than one project, it would be useful to us to know how much time you spent working on each project. You can do it by separating the Tracing task by project:

- Tracing: project user22
- Tracing: project user22-1
- Tracing: project user22-2

**Post-Task Survey**

The experiment is considered completed after you have submitted the final RTM and answered the post-experimental survey.

The copy of the survey will be provided to you upon declared completion of the tracing task. When you have completed the tracing task,

- Email the final RTM to dcuddeba@calpoly.edu.
- Upon receiving your RTM, you will be emailed a link to the post-experiment survey.
• Hand in your time card to your professor.

Checklist

How do you know when you’ve completed the experiment? You’re task is complete when you’ve finished the items on the following checklist:

1. Emailed final RTM to dcuddeba@calpoly.edu with your userID as the file name.

2. Submitted a post-experiment survey.

3. Handed in your time card to your professor.

Thank you very much for your participation!
Appendix B

Training Dataset

B.1 Functional Requirements

FM-1

Upon its start, the program shall print information about its purpose (moon landing game) and author:

********** Moon Landing Program v. 1.0 ***************

Written by: Alexander Dekhtyar

FM-2

After that the main loop of the program starts. At the beginning of each loop, the program shall display a text menu with three choices and prompt the user to make a selection. The menu choices are 1. Play Game, 2. Instructions, 3. Quit:

Please select an action:

1. Play Game
2. Instructions
3. Quit

Your choice :>>

(Note, there is an empty line between the Written by: line of the output and the Please select an action: line. Note also that :>> is the prompt used in the program.)
Your program shall accept user input, indicating the desired action. Acceptable user inputs are integers 1,2,3. If any other integer number is entered, the program shall simply output the main menu and the prompt again:

********** Moon Landing Program v. 1.0 **********
Written by: Alexander Dekhtyar

Please select an action:
1. Play Game
2. Instructions
3. Quit

Your choice :>>8

Please select an action:
1. Play Game
2. Instructions
3. Quit

Your choice :

Your program is not responsible for handling any other type of input (e.g., if a character or a string has been entered instead of an integer).

If the user chooses Quit, i.e., enters 3 the program shall print a goodbye message and exit.

Please select an action:
1. Play Game
2. Instructions
3. Quit

Your choice :>>3

Bye ! Thanks for playing Moon Landing

If the user chooses Instructions, i.e., enters 2 the instructions on how to play the game shall be printed. After that the program shall loop back to the menu. The text of the instructions is as follows:

Please select an action:
1. Play Game
2. Instructions
3. Quit

Your choice :>>2

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
You are to land a spaceship on the moon.
You control the speed of your spaceship by telling it how much fuel has to be burned every second.
5 units of fuel cancel the Moon gravity, more - will decelerate your ship.
Do not run out of fuel.
To land successfully your velocity should not exceed 10 feet/sec
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

Please select an action:
1. Play Game
2. Instructions
3. Quit

Your choice :>>
If the user chooses Play Game, i.e., enters 1, the game shall commence. See requirements below for the in-game specifications. After the game is over, the program shall loop back to the menu.

During the game initialization stage, the program shall set the initial values for the distance to the Moon, amount of fuel left and ship velocity. These parameters were specified above.

At the beginning of each game loop iteration, your program shall display current status of the game. Current status includes:

i. time elapsed since the beginning of the landing (in seconds);
ii. distance left to the Moon (in feet);
iii. amount of fuel left (units);
iv. current velocity (feet/sec).

After the status is displayed, the program shall prompt the user to enter amount of fuel to be burned (injected) over the next second. The program shall read the amount of fuel (an int value).

The amount of fuel entered by the user shall be tested: it shall not be negative and it cannot exceed amount of fuel left. The prompt for the amount of fuel to burn shall be repeated and a new value read until a valid value had been entered.

Once a valid amount of fuel is read the program computes the new velocity of the ship and the new distance to the moon. Use the formulas above to compute this.
B.2 System Tests

TC-1

Steps
------------------------
1. Start Moon Lander.

Output
------------------------
Please select an action:
1. Play Game
2. Instructions
3. Quit

Your choice :>

TC-2

Steps
------------------------
1. Start Moon Lander.
2. At the menu, type 3 and hit enter.

Output
------------------------
Bye ! Thanks for playing Moon Landing

TC-3

Steps
------------------------
1. Start Moon Lander.
2. At the menu, type 2 and hit enter.

Output
------------------------
* * * * * * * * The Game * * * * * * * * * * * * * * * * *
You are to land a spaceship on the moon. You control the speed of your spaceship by telling it how much fuel has to be burned every second. 5 units of fuel cancel the Moon gravity, more will decelerate your ship. Do not run out of fuel. To land successfully your velocity should not exceed 10 feet/sec
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Please select an action:
1. Play Game
2. Instructions
3. Quit

Your choice :>
TC-4

Steps
------------------------
1. Start Moon Lander.
2. At the menu, type 1 and hit enter.

Output
------------------------
Time Elapsed: 0 seconds
Distance: 500 feet
Fuel: 120
Velocity: 50 feet/sec

TC-5

Steps
------------------------
1. Start Moon Lander.
2. At the menu, type 1 and hit enter.
3. Burn 10 units of fuel: type 10 and hit enter.

Output
------------------------
Time Elapsed: 1 seconds
Distance: 427.5 feet
Fuel: 110
Velocity: 45 feet/sec
Appendix C

Experimental Dataset

C.1  Requirements

2.1.1

Formats source code in the BlueJ editor window on demand (does not format on the fly).

2.1.2

The user can edit the Style Preferences through the Style Preference GUI interface.

2.1.3

The user may specify or use default Style Preferences.
2.1.4

Presents the user the option to choose from preset Style Preferences (such as the GNU or JAVA styles) as well as user's Style Preferences.

2.1.5

Immediately after ChangeStyle formats the code, the user may restore the file to its state right before the format occurred.

2.1.6

Support all standard BlueJ class types {Class, Abstract Class, Interface, Applet, Unit Test, Enum}.

2.1.7

Available External Documentation (user manual).

2.1.10

Has the ability to format multiple files at the same time (i.e. all source code in a package.) Also have the ability to format just one file at a time.

2.1.12

ChangeStyle will only format compliable code.
2.1.13

ChangeStyle only formats files ending in '.java'.

2.1.14

Every panel in the settings dialog will have a personal help link linking to the corresponding section in the Jalopy User Manual, if the section is available.

2.1.15

Customized Conventions ChangeStyle will allow the user to modify predefined conventions and save them as customized conventions that will appear in the conventions drop down box along with the original predefined conventions.

2.1.16

Batch Undo ChangeStyle implements Batch Undo, which presents the user with a dialog with a selectable list of files which can be undone, while displaying files which cannot be undone in grey.

2.1.17

Editor Update Changes made by ChangeStyle's formatting are visible immediately in the editor window, and when the Undo button is selected in the editor window, the editor window shows the undone code.

2.1.18

Import/Export The import and export convention functionality for ChangeStyle is located in the Preferences Panel and is designed based on the approved UI prototype.
2.1.19

Headers/Footers ChangeStyle allows the user to replace Headers and Footers containing some user-specified text in the corresponding panels and offers a help link to a new section in the User Manual describing Headers and Footers in more detail.

2.1.20

Status Panel Every time a class or project is formatted by ChangeStyle, the 'status panel' at the bottom of BlueJ's window displays 'Formatting... Done.' When the undo feature is selected, the window will display 'Undo Formatting... Done.'

2.1.21

Preference Panel When the users clicks, the 'Edit...' button, no matter how many times, the Preferences panel only appears once.

3.0.1

Minimum: 64MB main memory, Pentium II processor or equivalent Recommended: 128MB main memory, 400MHz Pentium III processor or above

3.0.2

ChangeStyle does not take more than 10 seconds to format 1000 lines of code.

3.0.3

J2SE 1.6.0 (Java 2 SDK version 1.6.0) or newer must be installed.
3.0.4

ChangeStyle runs on BlueJ 2.2 or newer

3.0.5

ChangeStyle runs on Windows XP (SP2) and Mac OSX and Linux

3.0.6

ChangeStyle formats compiled code according to Jalopy's formatting convention standards.

3.0.9

The user can add and modify Style Preferences through the UI Portability

3.0.10

ChangeStyle is an extension to BlueJ which only runs on the Java Runtime Environment Reliability

3.0.11

ChangeStyle's Mean Time Between Defects (MTBD) is greater than 30 formats.
3.0.12

ChangeStyle does not format on uncompilable .java files.

3.0.14

ChangeStyle does not add any new expressions to the formatted code.

3.0.16

A Computer Science student can download and install ChangeStyle in less than 30 minutes on a computer that has BlueJ already installed.

3.0.17

A Computer Science student can format a .java file in BlueJ that has ChangeStyle already installed in less than 5 minutes.

3.0.18

Formatting a file using the currently set Style Convention takes less than 4 clicks.
C.2 System Tests

TC-1

Purpose:
Test that a format action properly updates the editor of a BClass and enables/disables the undo menu item properly.

Procedure:
- Download the test project directory from SVN.
- Open BlueJ with ChangeStyle installed.
- Open the test project.
- Reset ChangeStyle to DalbeyStyle through the Extensions Tab in BlueJ’s Preferences dialog.
- Follow the steps below.

Test Data:

<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double-click on Test1</td>
<td>The BlueJ editor window appears with Undo button disabled</td>
<td></td>
</tr>
<tr>
<td>Click back to main BlueJ window</td>
<td>BlueJ main window gains focus</td>
<td></td>
</tr>
<tr>
<td>Right-click Test1</td>
<td>Class context menu appears with ChangeStyle as an option</td>
<td></td>
</tr>
<tr>
<td>Click on ChangeStyle</td>
<td>Sub-menu appears with Format enabled and Undo disabled</td>
<td></td>
</tr>
<tr>
<td>Click on Format</td>
<td>Text in editor window is formatted (to match Test1.txt) and the Undo button is enabled. May take effect on next step</td>
<td></td>
</tr>
<tr>
<td>Click on editor window</td>
<td>Editor gains focus</td>
<td></td>
</tr>
<tr>
<td>Open a command window or terminal</td>
<td>diff &lt;TestDir&gt;/Test1.java &lt;TestDir&gt;/Test1.txt (UNIX)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fc &lt;TestDir&gt;/Test1.java &lt;TestDir&gt;/Test1.txt (Windows)</td>
<td></td>
</tr>
</tbody>
</table>


TC-2

Purpose:
To show how the user can change between different Style Preferences as well as user’s Style Preferences.

Procedure:
- Open BlueJ with ChangeStyle installed.
- Create a new project.
- Create a new Class
- Follow the steps below.

Test Data:

<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>In main BlueJ window, click the ‘Tools’ located in the menu across the top of the screen</td>
<td>A pull-down menu appears</td>
<td></td>
</tr>
<tr>
<td>Select and click ‘Preferences...’</td>
<td>A pop up window appears giving information about preferences</td>
<td></td>
</tr>
<tr>
<td>Find and click on the ‘Extensions’ tab</td>
<td>The tab should change and you should see different detail sections for each extension</td>
<td></td>
</tr>
<tr>
<td>Under the ChangeStyle detail section, click the button labeled ‘Edit...’</td>
<td>A ‘Custom Settings’ pop-up window and a ‘Preview’ pop-up window should appear</td>
<td></td>
</tr>
<tr>
<td>Looking over the ‘Custom Settings’ pane, you can find under the ‘Braces’ default option, ‘Styles’ option, different preset styles to choose from or you can go through each option and create your own style preferences</td>
<td>After selecting ‘Ok’ when a format is invoked, the code should be formatted according to the chosen style</td>
<td></td>
</tr>
</tbody>
</table>
TC-3

**Purpose:**
Test that format works on each BlueJ class type.

**Procedure:**
* Download the test project directory from SVN.
* Open BlueJ with ChangeStyle installed.
* Open the test project.
* Use the Tools/Preferences menu to select the Sun Style convention.
* Follow the steps below.

**Test Data:**
<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on the Compile button</td>
<td>All classes are compiled</td>
<td></td>
</tr>
<tr>
<td>Using the Tools menu click on ChangeStyle Sub-menu appears with Format Entire Project enabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Click on Format Entire Project</td>
<td>The classes are formatted</td>
<td></td>
</tr>
<tr>
<td>Now try to right click on the paper icon in the environment</td>
<td>You will notice that a menu doesn't pop up offering the formatting option, since the file is a .txt file</td>
<td></td>
</tr>
<tr>
<td>Use diff or fc to confirm the format from a terminal or command prompt</td>
<td>diff &lt;testProject&gt;/<em>.java &lt;testProject&gt;/</em>.txt No differences should appear</td>
<td></td>
</tr>
</tbody>
</table>

TC-4

**Purpose:**
Show where to locate available external documentation.

**Procedure:**
* Open BlueJ with ChangeStyle installed.
* Create a new project.
* Reset Jalopy to its default settings through the Extensions Tab in BlueJ’s Preferences dialog.
* Follow the steps below.

**Test Data:**
<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>In main BlueJ window, click the help tab at the top of the screen</td>
<td>A pull-down menu appears</td>
<td></td>
</tr>
<tr>
<td>Select and click 'Installed Extensions'</td>
<td>A pop up window appears giving information about all the installed extensions</td>
<td></td>
</tr>
<tr>
<td>Click on the bubble with the question mark inside of it (next to the status column of the ChangeStyle installed extension)</td>
<td>A pop up window should occur giving Extension Details</td>
<td></td>
</tr>
<tr>
<td>Click on the link provided next to the 'More Information At:' label</td>
<td>A web browser should pop up, open to Luna’s Home Page</td>
<td></td>
</tr>
<tr>
<td>Find and click on the 'User Manual' link pertaining to your version of the ChangeStyle extension</td>
<td>Webpage opened to the User Manual</td>
<td></td>
</tr>
</tbody>
</table>
**TC-5**

**Purpose:**
Tests the formatter for safety when handling uncompiled code.

**Procedure:**
1. Download the test project directory from SVN.
2. Open BlueJ with ChangeStyle installed.
3. Open the test project.
4. Reset ChangeStyle to DalbeyStyle through the Extensions Tab in BlueJ’s Preferences dialog.
5. Follow the steps below.

<table>
<thead>
<tr>
<th>Test Data:</th>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Double-click on Test1</td>
<td>The editor window appears for Test1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Click anywhere in the text area of the editor</td>
<td>A blanking vertical cursor appears where you clicked.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Press the Space Bar on the keyboard</td>
<td>A space is inserted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Press the Backspace key on the keyboard</td>
<td>The added space is deleted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Click on the main BlueJ window</td>
<td>BlueJ main window gains focus and Test1 appears shaded (uncompiled)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Right-click on Test1</td>
<td>Class context menu appears</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Click on ChangeStyle</td>
<td>Sub-menu appears</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Verify that Format is disabled</td>
<td>The Format option should appear grey with the text (not compiled)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. Click on the Format option</td>
<td>Nothing should happen and the menu should stay visible</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. Click on the Compile button</td>
<td>Test1 is compiled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. Repeat steps 6 &amp; 7</td>
<td>Format should be enabled now</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. Click on Format</td>
<td>the class is formatted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. Use diff or fc to confirm the format from a terminal or command prompt</td>
<td>diff &lt;testProject&gt;/<em>.java &lt;testProject&gt;/</em>.txt No differences should appear</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TC-6**

**Purpose:**
Verify that ChangeStyle will operate on a computer meeting the system specifications.

**Procedure:**
1. Look up the computers memory and the processor information.
2. Follow the steps below.

<table>
<thead>
<tr>
<th>Test Data:</th>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look up main memory</td>
<td>Minimum 64MB of memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Look up processor</td>
<td>Pentium II or equivalent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify both processor and main memory meet or exceed minimum requirements</td>
<td>Pentium II or equivalent, 64MB of memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run BlueJ</td>
<td>BlueJ opens.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify ChangeStyle is installed: Help-&gt;Installed Extensions</td>
<td>ChangeStyle should be displayed under Installed Extensions.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TC-7**

**Purpose:**
Verify that ChangeStyle takes 1 seconds or less to format 100 lines of code.

**Procedure:**
1. Format a class that has about than 100 lines of code. Since comments can be formatted as well they are included.
2. Follow the steps below.

<table>
<thead>
<tr>
<th>Test Data:</th>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start BlueJ</td>
<td>BlueJ window open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open testcase07.java</td>
<td>Class should appear in BlueJ window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open Timer</td>
<td>Timer should be at 00:00:00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start Timer and Format testcase07.java</td>
<td>Confirm button should appear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop Timer when confirm button appears</td>
<td>Timer should display less than 00:00:01.00 or 1 second.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TC-8

Purpose:
- Verify that the computer has J2SE 1.5.0 (Java 2 SDK version 1.5.0) or newer must be installed.
- Verify that BlueJ 2.2 or newer.
- Verify that ChangeStyle operates when requirements are met.
- Follow the steps below:

Test Data:

<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify Java Runtime Environment meets requirements.</td>
<td>J2SE 1.5.0 (Java 2 SDK version 1.5.0) or newer</td>
<td></td>
</tr>
<tr>
<td>Verify BlueJ version number meets requirements</td>
<td>BlueJ 2.2 or newer</td>
<td></td>
</tr>
<tr>
<td>Run BlueJ</td>
<td>BlueJ UI opens</td>
<td></td>
</tr>
<tr>
<td>Verify ChangeStyle is installed: Help-&gt;Installed Extensions</td>
<td>ChangeStyle should be displayed in Installed Extensions</td>
<td></td>
</tr>
</tbody>
</table>

TC-9

Purpose:
- Checks to see if ChangeStyle was downloaded and works for Windows XP (SP2), Mac OSX and Red Hat Linux.

Procedure:
Follow the steps below:

Test Data:

<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download ChangeStyle as a BlueJ extension</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open BlueJ</td>
<td>Class should appear in BlueJ window</td>
<td></td>
</tr>
<tr>
<td>Open Help</td>
<td>Help window will open</td>
<td></td>
</tr>
<tr>
<td>Click on Installed Extensions</td>
<td>BlueJ:Installed Extensions window will open</td>
<td></td>
</tr>
<tr>
<td>window will say loaded ChangeStyle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format testCase09.java</td>
<td>Confirm window will open, saying class has been formatted.</td>
<td></td>
</tr>
<tr>
<td>Do this for all System, Windows XP (SP2), Mac OSX and Red Hat Linux</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TC-10

Purpose:
- To verify that ChangeStyle requires that a Java Runtime Environment be installed to run.

Procedure:

- Verify that the computer you are working on does not have a Java Runtime Environment installed. If there is one installed, then uninstall it.
- Follow the steps below.

Test Data:

<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attempt to download and install BlueJ using the instructions here:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="http://www.bluej.org/download/download.html">http://www.bluej.org/download/download.html</a></td>
<td>BlueJ should not be able to install.</td>
<td></td>
</tr>
</tbody>
</table>
TC-11

Purpose:
To verify that ChangeStyle’s mean time between defects is greater than 30 formats.

Procedure:
- Open BlueJ. If BlueJ is not installed follow the instructions here: http://www.bluej.org/download/download.html to download and install it.
- Check if ChangeStyle is installed by clicking “Help->Installed Extensions” on the menu panel and verifying that “ChangeStyle” is on the list of extensions and is “loaded”. If ChangeStyle is not installed follow the instructions here: http://wiki.csc.calpoly.edu/Luna/wiki/download to download and install it.
- Follow the steps below.

Test Data:
<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click “Project-&gt;New Project” in the BlueJ menu panel.</td>
<td></td>
<td>A Create Project Window should appear.</td>
</tr>
<tr>
<td>Name the Project “DefectTest” and click “Create”.</td>
<td></td>
<td>DefectTest A new Project Window with no Class boxes in it should appear.</td>
</tr>
<tr>
<td>Click on “New Class” on BlueJ’s left side panel.</td>
<td></td>
<td>A Create New Class Window should appear.</td>
</tr>
<tr>
<td>Name the class “DefectTestClass” and click “OK”.</td>
<td></td>
<td>A “box” with the name “DefectTestClass” in the Project Window should appear.</td>
</tr>
<tr>
<td>Right-Click the “DefectTestClass” box and click “Compile”.</td>
<td></td>
<td>The box should appear plain, not having diagonal lines through it.</td>
</tr>
<tr>
<td>Right-Click the “DefectTestClass” box, scroll down to “ChangeStyle”, and click “Format”</td>
<td></td>
<td>If BlueJ freezes or closes prematurely record that a defect occurred and the time.</td>
</tr>
<tr>
<td>Click on “Tools-&gt;ChangeStyle-&gt;Format” in the BlueJ menu panel.</td>
<td></td>
<td>If BlueJ freezes or closes prematurely record that a defect occurred and the time.</td>
</tr>
<tr>
<td>Repeat the previous two steps 30 more times. (Formatting the code)</td>
<td></td>
<td>After 30 or more formats record how many total defects occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defects &lt;= 1</td>
</tr>
</tbody>
</table>

TC-12

Purpose:
To verify that ChangeStyle does not format on uncompilable .java files.

Procedure:
- Open BlueJ. If BlueJ is not installed follow the instructions here: http://www.bluej.org/download/download.html to download and install it.
- Check if ChangeStyle is installed by clicking “Help->Installed Extensions” on the menu panel and verifying that “ChangeStyle” is on the list of extensions and is “loaded”. If ChangeStyle is not installed follow the instructions here: http://wiki.csc.calpoly.edu/Luna/wiki/download to download and install it.
- Follow the steps below.

Test Data:
<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click “Project-&gt;New Project” in the BlueJ menu panel.</td>
<td></td>
<td>A Create Project Window should appear.</td>
</tr>
<tr>
<td>Name the Project “FormatTest” and click “Create”.</td>
<td></td>
<td>FormatTest A new Project Window with no Class boxes in it should appear.</td>
</tr>
<tr>
<td>Click on “New Class” on BlueJ’s left side panel.</td>
<td></td>
<td>A Create New Class Window should appear.</td>
</tr>
<tr>
<td>Name the class “FormatTestClass” and click “OK”.</td>
<td></td>
<td>A “box” with the name “FormatTestClass” in the Project Window should appear.</td>
</tr>
<tr>
<td>Double click on the “FormatTestClass” box.</td>
<td></td>
<td>A Class Editor Window should appear.</td>
</tr>
<tr>
<td>Type “not valid code” as the first line of the Class and close the Editor Window.</td>
<td></td>
<td>Not valid code. The Class Editor Window should disappear.</td>
</tr>
<tr>
<td>Right-Click the “FormatTestClass” box and scroll down to “ChangeStyle”.</td>
<td></td>
<td>A grayed-out button that says “Format (not compiled)” should appear.</td>
</tr>
<tr>
<td>Click on “Tools-&gt;ChangeStyle” in the BlueJ menu panel.</td>
<td></td>
<td>A grayed-out button that says “Format (not compiled)” should appear.</td>
</tr>
</tbody>
</table>
TC-13

Purpose:
* Verify the formatting function will not change the source code in a functional way.

Procedure:
* Open BlueJ.
* Create a new project containing the following class:
  ```java
  TestCase3014.java
  ```
* Follow the steps below:

Test Data:
<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start BlueJ</td>
<td>The BlueJ main window appears.</td>
<td>The new project window is displayed.</td>
</tr>
<tr>
<td>Open the project containing TestCase3014.java</td>
<td>Run the class.</td>
<td>&quot;Hello World!&quot;</td>
</tr>
<tr>
<td>Change the brace style to GNU.</td>
<td>Right click on the class TestCase3014 and select ChangeStyle</td>
<td>The class should now be displayed as uncompiled.</td>
</tr>
<tr>
<td>Compile and run the class.</td>
<td></td>
<td>&quot;Hello World!&quot;</td>
</tr>
<tr>
<td>Open the Editor window and verify no new expressions were added to the code.</td>
<td>The formatting should have only changed, no new expressions displayed.</td>
<td></td>
</tr>
</tbody>
</table>

File: TestCase3014.java
```
public class TestCase3014 {
    public static void main(String[] args) {
        System.out.println("Hello World!");
        for(int i=1; i<=10; i++) {
            System.out.println(i);
        }
    }
}
```
TC-14

**Purpose:**
- Verify the user can install ChangeStyle in less than 30 minutes.

**Procedure:**
- Open the ChangeStyle website.
- Download the most recent ChangeStyle jar file.
- Follow the installation instructions found on the Luna website.
- Verify ChangeStyle is installed and detected by BlueJ.

**Test Data:**

<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Luna Software's website. <a href="http://wiki.csc.calpoly.edu/Luna/wiki">http://wiki.csc.calpoly.edu/Luna/wiki</a></td>
<td>The main wiki page for Luna Software is displayed.</td>
</tr>
<tr>
<td>Click</td>
<td>the link &quot;Download ChangeStyle&quot;.</td>
<td>The download page is displayed.</td>
</tr>
<tr>
<td>Download</td>
<td>the latest version and follow the installation instructions listed</td>
<td>ChangeStyle should now be installed.</td>
</tr>
<tr>
<td>on</td>
<td>on the bottom of the page</td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>BlueJ.</td>
<td>The main BlueJ window is displayed.</td>
</tr>
<tr>
<td>Click</td>
<td>on Help -&gt; Installed Extensions.</td>
<td>The Installed Extensions window is displayed.</td>
</tr>
<tr>
<td>Verify</td>
<td>ChangeStyle is listed and the status displays 'loaded.'</td>
<td></td>
</tr>
<tr>
<td>Calculate</td>
<td>the time spent installing.</td>
<td>The install time should be less than 30 minutes.</td>
</tr>
</tbody>
</table>

TC-15

**Purpose:**
- Verify the user can format the code in under 5 minutes.
- Verify the user can format in as little as two clicks.

**Procedure:**
- Open BlueJ.
- Create a new project containing the following class:
  ```java
  public class !TestCase3017 {
      public static void main(String[] args) {
          System.out.println("Hello World!");
          for(int i=0; i<10; i++) {
              System.out.println(i);
          }
      }
  }
  ```

**Test Data:**

<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>BlueJ.</td>
<td>The BlueJ main window appears.</td>
</tr>
<tr>
<td>Open</td>
<td>the project containing !TestCase3017.java.</td>
<td>The new project window is displayed.</td>
</tr>
<tr>
<td>Right</td>
<td>click on the class !TestCase3017 and select ChangeStyle</td>
<td>The class is now formatted and should be displayed as uncompiled.</td>
</tr>
<tr>
<td>Verify</td>
<td>the procedure took less than 5 minutes.</td>
<td></td>
</tr>
</tbody>
</table>

File: !TestCase3017.java

```java
public class !TestCase3014 {
    public static void main(String[] args) {
        System.out.println("Hello World!");
        for(int i=0; i<10; i++) {
            System.out.println(i);
        }
    }
}
```
TC-17

Purpose:
* User can add/modify Style Preferences through UI.

Procedure:
* Download and install "changestyle.jar" from the LUNA Wiki Page.
* Create a new file called "JDalbeyConvention.xml" with the text from case #16.
* Create a new BlueJ Project called "Test Case MARK".
* Create a new Class called "TestClass".
* Edit "TestClass" (see case #18).
* Go to Tools > Preferences...
* Go to the [Extensions] tab.
* Click [Edit...].
* Navigate to the General section.
* Type "New Convention" in the "Name:" field.
* Type "New (Test) Convention" in the "Description:" field.
* Navigate to the Printer > Indentation section.
* Set the "General indent:" field to "8".
* Click [Apply].
* Navigate to the General section.
* Click [Export...].
* Click [Browse...].
* Type "New Convention" in the "File name:" field.
* Click [OK].
* Click [Import...].
* Click [Browse...].
* Find the file "JDalbeyConvention.xml", click on it and then click [OK].
* Click [OK].
* Click [Import...].
* Click [Browse...].
* Find the file "NewConvention.xml", click on it and then click [OK].
* Click [OK].
* -- compare expected output #1 below --
* Click [OK].
* Click [OK].
* Close BlueJ.

Test Data:
Expected Output #1

General Section
Name: JDalbey Convention
Description: JDalbey 305/308/309 Coding Convention

Printer > Indentation Section (General Tab)
General indent: 4
JDalbeyConvention.xml

see test case 16
Purpose:

* Verify that ChangeStyle formats code properly

Procedure:

* Download and install "changeStyle.jar" from the LUNA Wiki Page.
* Create a new file called "JDalbeyConvention.xml"
* Edit "JDalbeyConvention.xml" (see below).
* Create a new BlueJ Project called "Test Case MARK".
* Create a new Class called "TestClass".
* Go to Tools > Preferences...
* Go to the [Extensions] tab
* Click [Edit...]
* Navigate to the General section.
* Click on the Import button
* Click the Browse button
* Choose JDalbeyConvention.xml
* Click the OK Button
* Navigate to the Printer > Indentation section.
* Set the "General indent:" field to "8".
* Navigate to the Printer > Braces section.
* Click the box next to "Sun Java style".
* Click [Apply].
* Click [OK].
* Click [OK].
* Right-click "TestClass".
* Click Compile.
* Right-click "TestClass".
* Click ChangeStyle > Format.
* -- compare expected output #1 below --
* Right-click "TestClass".
* Click Open Editor.
* -- compare expected output #2 below --
* Close the "TestClass" Editor.
* Close BlueJ

Test Data:

Expected Output #1:
The "TestClass" icon should have 'hash marks' indicating it is not compiled. No pop-up messages should appear.

Expected Output #2:

```java
public class TestClass {
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class TestClass
     */
    public TestClass() {
        // initialise instance variables
        x = 0;
    }
}
```

JDalbeyConvention.xml

see case 16
Appendix D

Results

D.1 Extra Graphs

![Graph showing results of all participants](image)

Figure D.1: Results of all participants.
Figure D.2: Analyst effort vs. starting and ending size of the RTM.
Figure D.3: Analyst effort vs. size of the (a) initial RTM and (b) final RTM.
Figure D.4: Analyst effort vs. the change in size of the RTM.

Figure D.5: Analyst effort vs. the recall and precision of the initial RTM.
Figure D.6: $f_2$-measure of the final RTM vs. the size of the (a) initial and (b) final RTM.
### Tables

<table>
<thead>
<tr>
<th>ID</th>
<th>Initial</th>
<th>Final</th>
<th>Added</th>
<th>Removed</th>
<th>Kept</th>
<th>Missed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33</td>
<td>30</td>
<td>0</td>
<td>3</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>33</td>
<td>30</td>
<td>0</td>
<td>3</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td>30</td>
<td>0</td>
<td>3</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>28</td>
<td>30</td>
<td>4</td>
<td>2</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>30</td>
<td>7</td>
<td>2</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>29</td>
<td>1</td>
<td>3</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>23</td>
<td>28</td>
<td>6</td>
<td>1</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>28</td>
<td>11</td>
<td>1</td>
<td>17</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>25</td>
<td>27</td>
<td>4</td>
<td>2</td>
<td>23</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>27</td>
<td>4</td>
<td>2</td>
<td>23</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>33</td>
<td>26</td>
<td>0</td>
<td>7</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>21</td>
<td>26</td>
<td>9</td>
<td>4</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>33</td>
<td>25</td>
<td>0</td>
<td>8</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>25</td>
<td>25</td>
<td>4</td>
<td>4</td>
<td>21</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>18</td>
<td>23</td>
<td>6</td>
<td>1</td>
<td>17</td>
<td>9</td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>17</td>
<td>5</td>
<td>6</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>17</td>
<td>7</td>
<td>16</td>
<td>11</td>
<td>2</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>12</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>19</td>
<td>7</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>22</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td>21</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>26</td>
</tr>
<tr>
<td>22</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>33</td>
</tr>
</tbody>
</table>

Table D.1: Frequency of actions taken for true links.
\begin{tabular}{cccccccc}
ID & Initial & Final & Added & Removed & Kept & Missed \\
\hline
24 & 30 & 24 & 0 & 6 & 24 & 3 \\
25 & 24 & 20 & 3 & 7 & 17 & 6 \\
26 & 20 & 16 & 3 & 7 & 13 & 10 \\
27 & 16 & 16 & 3 & 3 & 13 & 14 \\
28 & 0 & 15 & 15 & 0 & 0 & 18 \\
29 & 1 & 13 & 12 & 0 & 1 & 20 \\
30 & 12 & 12 & 6 & 6 & 6 & 15 \\
31 & 9 & 12 & 4 & 1 & 8 & 20 \\
32 & 9 & 11 & 4 & 2 & 7 & 20 \\
33 & 22 & 10 & 1 & 13 & 9 & 10 \\
34 & 12 & 9 & 3 & 6 & 6 & 18 \\
35 & 17 & 8 & 1 & 10 & 7 & 15 \\
36 & 17 & 8 & 2 & 11 & 6 & 14 \\
37 & 9 & 8 & 4 & 5 & 4 & 20 \\
38 & 27 & 7 & 0 & 20 & 7 & 6 \\
39 & 12 & 7 & 2 & 7 & 5 & 19 \\
40 & 0 & 7 & 7 & 0 & 0 & 26 \\
41 & 17 & 5 & 0 & 12 & 5 & 16 \\
42 & 11 & 5 & 0 & 6 & 5 & 22 \\
43 & 9 & 5 & 1 & 5 & 4 & 23 \\
44 & 9 & 5 & 1 & 5 & 4 & 23 \\
45 & 0 & 5 & 5 & 0 & 0 & 28 \\
46 & 0 & 5 & 5 & 0 & 0 & 28 \\
47 & 18 & 4 & 0 & 14 & 4 & 15 \\
48 & 17 & 4 & 1 & 14 & 3 & 15 \\
49 & 17 & 4 & 1 & 14 & 3 & 15 \\
50 & 17 & 4 & 0 & 13 & 4 & 16 \\
51 & 17 & 4 & 0 & 13 & 4 & 16 \\
52 & 9 & 4 & 1 & 6 & 3 & 23 \\
53 & 9 & 4 & 0 & 5 & 4 & 24 \\
54 & 9 & 4 & 1 & 6 & 3 & 23 \\
55 & 9 & 4 & 1 & 6 & 3 & 23 \\
56 & 9 & 4 & 2 & 7 & 2 & 22 \\
57 & 5 & 4 & 2 & 3 & 2 & 26 \\
58 & 0 & 4 & 4 & 0 & 0 & 29 \\
\hline
\end{tabular}

Table D.2: Frequency of actions taken for false links.