**Jane Huffman Hayes • Alex Dekhtyar •
Senthil Sundaram • Ashlee Holbrook •
Sravanthi Vadlamudi • Alain April**

REquirements Tracing On target (RETRO): Improving Software Maintenance through Traceability Recovery

Abstract A number of important tasks in software maintenance require an up-to-date requirements traceability matrix (RTM): change impact analysis, determination of test cases to execute for regression testing, etc. Generation and maintenance of RTMs is tedious and error-prone, and hence it is often not done. In this paper, we present RETRO (*REquirements TRacing On-target*), a special-purpose requirements tracing tool. We discuss how RETRO automates the generation of RTMs and present the results of a study comparing manual RTM generation to RTM generation using RETRO. The study showed that RETRO found significantly more correct links than manual tracing and took only one third of the time to do so.

1 Introduction

Software maintenance is central to the mission of many organizations. It consumes a large part of the software lifecycle costs and there are billions of lines of code under maintenance in the world [23]. One of the hardest problems in software maintenance is to understand the program and to localize the program parts that should be modified to complete the maintenance task at hand. The problem can be serious when maintaining systems that have been evolved by many different individuals using agile methodologies that yield little documentation.

Software traceability is becoming recognized as a significant contributor to efficient software and system quality. However, as empirical studies and quality audits of industrial organizations have indicated, its practice and instrumentation is not always satisfactory. An explanation often stated to justify non-conformance of keeping the traceability links consistent is the process itself which is time consuming, error-prone, and labor-intensive.

In many industries, the software maintenance methodology requirements state that documented bi-directional traceability needs to be maintained over the entire life of the system. This facilitates software change impact analysis, reuse analysis, program comprehension, regression testing, etc. The main issue is that software maintainers find the update of the system documentation to be tedious and hence it is often neglected. To verify the accuracy of or to recreate a traceability matrix that is not well-maintained makes it necessary to create traceability links and matrices "after-the-fact." This activity is called traceability recovery.

In addition, the process of creating and maintaining a requirements tracing matrix (RTM) is time consuming and error-prone. The tools that are available to assist with tracing are aimed at developers who are creating the trace as they develop the system the first time. They do not readily support the maintenance of an RTM or after the fact generation of an RTM. Clearly there is a need for automation.

In this paper, we present RETRO, REquirements TRacing On-target, a tool that we have built to address the recovery of traceability for artifacts containing unstructured textual narrative. RETRO uses Information Retrieval (IR) and text mining methods to construct candidate traces. To date, it has been used to trace requirements and design documents [2,3] and collections of bug reports [4]. The tool has evolved from a research only tool-kit into a more industrial tool directed at verification and validation (V&V) analysts as well as maintainers in several countries. The tool consists of a set of information retrieval (IR) and text mining methods as well as a front-end that provides functionality for the analyst to use during the tracing process. Our work to date has largely focused on the quality of generated traces as a function of the information retrieval methods used [1,2,3].

We have begun to venture into an examination of how the analyst interacts with such a tool, how usable the tool is, and how this impacts the quality of the final traceability matrix. A preliminary result showed that the analysts were satisfied with the back-end, but wanted a better front-end [5,6]. We set about to address these concerns, and the latest version of RETRO was developed after a year-long effort of re-design and improvement to the front-end capabilities of RETRO. In this paper, we report on the study we undertook to evaluate the usability of the resulting front-end.

The paper is organized into six sections. Information retrieval for tracing is presented in Section 2. The tracing tool, RETRO, is presented in Section 3. The empirical study undertaken to assess RETRO is discussed in Section 4. Related work is presented in Section 5. Finally, conclusions and future work are presented in Section 6.

2 Information Retrieval for Tracing

Since [1], we have observed that Information Retrieval methods can be adopted and, if necessary, adapted for use in tracing textual artifacts. Indeed, a typical IR problem involves a document collection and a user information need (expressed in the form of a text query). The task is to find documents in the collection that are deemed relevant to the query. When two artifacts are traced to each other, elements of one of the artifacts serve as "documents" in the "document collection," while the elements of the other serve as queries. In particular, when forward tracing (from a parent artifact to a child artifact) is considered, low-level elements form the "collection" while high-level elements become the queries.

We have incorporated a number of different IR methods in RETRO. Our prior work [5,6] suggests that analyst satisfaction with the tool depends mostly on the features/functionality available through the GUI rather than on the IR methods used. This paper concentrates on the front-end functionality of RETRO, but for illustrative purposes we describe one of the methods, *vector space retrieval with tf-idf term weighting* and with *standard Rochio* feedback processing [7]. This method is the default tracing technique in RETRO.

Vector space retrieval methods are the bread-and-butter of Information Retrieval. These methods represent each document in the document collection and each query as a vector of keyword weights, where keywords (or terms) are the words found in the documents. In particular, each document and query are passed through a stop word removal procedure, removing words with no significant importance, such as "a," "and," "to," and "shall." After that, the remaining text is stemmed to ensure that words such as "information," "informational," and "informative" are treated as the same term [7]. The vocabulary of the document collection, $D = \{k_1,\ldots,k_N\}$, is formed as the union of all terms found in all documents. Each document, d_i, is then represented as a vector, $di=(w_{i1},\ldots,w_{iK})$ of term/keyword weights. Different term weighting schemes can be used to construct these vectors. The most popular scheme, *tf-idf*, uses the formula $w_{ij} = tf_{ij} \cdot \log\left(\frac{n}{df_j}\right)$, where tf_{ij}, called *term frequency* of keyword k_j in document d_i, is the normalized frequency of occurrence of k_j in d_i, while $\log\left(\frac{n}{df_j}\right)$, is called the *inverse document frequency* of term k_j. That is, term weight is proportional to how often the term is found in the document and inversely proportional to (the logarithm of) how often it is found in the entire collection. Given a document vector *d* and a similarly computed query vector *q*, the similarity between them is computed as the cosine of the angle between the two vectors:

$$sim(d,q) = \cos(d,q) = \frac{\sum_{j=1}^{N} d_j \cdot q_j}{\sum_{j=1}^{N} d_j^2 \cdot \sum_{j=1}^{N} q_j^2}.$$

It is well-known in IR that the quality of retrieval can be improved by using user relevance feedback, i.e., the information about relevance or irrelevance of specific retrieved documents provided by humans back to an IR system. RETRO includes support for relevance feedback. Relevance feedback techniques for vector-space methods work by adjusting the keyword weights of *query vectors* according to the feedback. Feedback consists of "relevant" and "irrelevant" qualifications for some of the documents retrieved in the previous step. More formally, for a query *q*, let D_q be a list of document vectors retrieved. The user feedback identifies two subsets in D_q: D_r of size *R* of documents relevant to *q* and D_{irr} of size *S* of irrelevant documents. D_r and D_{irr} are disjoint, but do not necessarily cover the entire set D_q. We use the Standard Rochio [7] feedback processing method:

$$q_{new} = \alpha q + \left(\frac{\beta}{r} \sum_{d_j \in D_r} d_j \right) - \left(\frac{\gamma}{s} \sum_{d_k \in D_{irr}} d_k \right).$$

Here, query q is adjusted by adding to its vector a vector consisting of the document vectors identified as relevant, and subtracting from it the sum of all document vectors identified as not relevant. The first adjustment is designed to potentially increase recall (defined below). The second adjustment can potentially increase precision (defined below). The constants α, β, γ in the formulas above can be adjusted in order to emphasize positive or negative feedback as well as the importance of the original query vector (in this paper, all values were set to 1). Once the query vectors have been recomputed, the selected IR algorithm is re-run with the modified query vectors. This cycle can be repeated until the user is satisfied with the results.

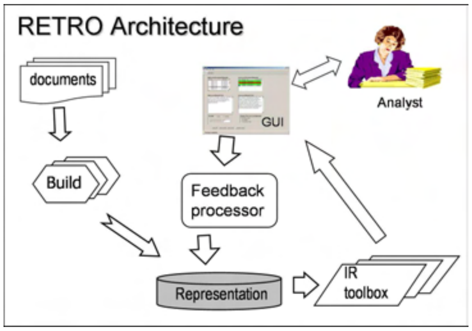


Fig. 1 Architecture of RETRO

Measuring the accuracy of IR methods. *Recall* and *precision* are two measures traditionally used to evaluate the accuracy of the results returned by IR methods. Informally, precision measures the percentage of retrieved documents that are relevant, while recall measures the percentage of relevant documents that were retrieved. More formally, if a document collection has *N* documents, *R* of which are relevant to query *q*, and an IR method retrieves n documents, *r* of which are relevant to *q*, then the precision and recall of the method on query *q* are defined as follows:

$$precision = \frac{r}{n}; \quad recall = \frac{r}{R}.$$

High recall of candidate link lists generated by IR methods used for traceability analysis means that the methods successfully discovered most of the links from the RTM, i.e., few errors of omission were committed. High precision of candidate link lists means that most of the links retrieved by the method were from the RTM, i.e., few errors of commission were committed. In our prior work [2,3,8], we argue that it is easier for an analyst to discover an error of commission, i.e., recognize that a retrieved candidate link is incorrect, than to recognize an error of omission, i.e., recognize that a valid link has not been reported.

3 RETRO

Originally, RETRO was designed as a nameless research toolbox of IR methods adopted and adapted, where needed, for requirements tracing. The name RETRO and the first front-end appeared only about one year after the original development. The purpose of the first front-end was simply to allow researchers to browse the results of tracing methods.

Over time, our view of RETRO has evolved. The concept of a special-purpose requirements tracing tool caught the eye of NASA and analysts working on NASA Independent Verification and Validation (IV&V) projects. Our first attempts to use RETRO in such contexts, as well as our work on new tracing methods [5,6,9], lead us to the observation that IV&V analysts were content with the RETRO backend, but would like to see the front-end of RETRO implement a wider range of facilities for tracing. Our most recent effort has lead to the complete redevelopment of the RETRO front end and development of additional functionality.

3.1 Evolution of RETRO

The first version of the current RETRO GUI (RETRO 2.0) was developed with basic functionalities that allowed an analyst to work with the IR methods, to view the results, and then to provide some feedback. The version didn't provide support for viewing the final trace or for searching for any links that may have been omitted by the IR methods. Also, the basic functionalities provided were not easy to use. As this version was developed with minimal options, it posed problems for users such as lack of functionality and lack of usability.

The next version of RETRO, 2.3, added the functionality required for tracing a project and also fixed problems from the first version. This version allowed users to reject links that were not correct (errors of commission, i.e, errors made by the IR methods in retrieving the candidate linked lists), but did not allow the user to report errors of omission (links missed by the IR methods).

RETRO, 2.5, had additional support for reporting errors of omission using a separate tab called 'Browse' which also provided support for manual tracing. This version also provided filtering functionality to allow the user to control the display of candidate links.

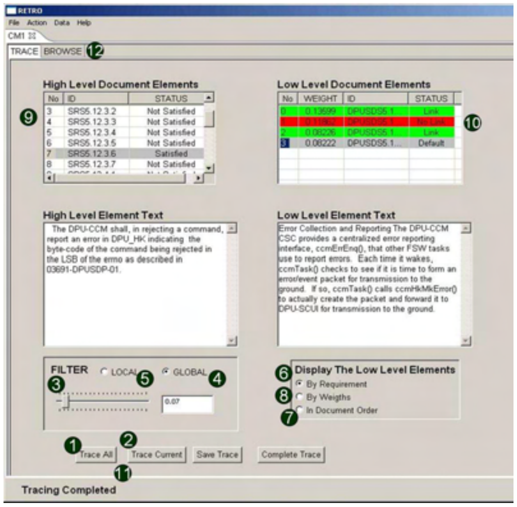
The next version of RETRO, 2.7, was developed to include functionality for the analyst to control the display of the requirements and to allow the analyst to view the completed projects in an easy to understand way. This version also added support for searching for keywords in the Browse tab. This version had some scalability issues and failed to work when large projects were loaded. In addition to addressing these issues, the final beta version of RETRO (2.N.N) added the ability to assess an existing RTM and also added enhanced functionality for filtering the display of candidate links.

3.2. Architecture of RETRO

Figure 1 shows the architecture of RETRO. The core part of RETRO consists of the IR Toolbox, the Feedback processing methods, and the GUI front end. In addition to this, methods for building representations of traced documents are included. At the present time, all components except for the GUI are written in C++, while the front end of RETRO was developed in Java using Eclipse's SWT GUI library. The components communicate with each other in one of two ways: (i) by changing the representation of the documents stored on disk, or (ii) by using XML files encoding candidate link lists and user feedback information. In particular, build methods and the feedback processor change the representation of the documents on disk, while the toolbox methods encode their results in an XML file read by the GUI. The GUI solicits user feedback, and based on it, modifies the XML file, which it then passes to the feedback processor for a new round of tracing.

3.3. Functionality of RETRO

The version of RETRO described here, RETRO 2.5, has been



High Level Document Elements

No	ID	STATUS
3	SRS5.12.3.2	Not Satisfied
4	SRS5.12.3.3	Not Satisfied
5	SRS5.12.3.4	Not Satisfied
6	SRS5.12.3.5	Not Satisfied
7	SRS5.12.3.6	Satisfied
8	SRS5.12.3.7	Not Satisfied

Low Level Document Elements

No	WEIGHT	ID	STATUS
0	0.13699	DPUSDS5.1...	Link
1	0.11862	DPUSDS5.1...	No Link
2	0.08226	DPUSDS5.1...	Link
3	0.08222	DPUSDS5.1...	Default

High Level Element Text

The DPU-CCM shall, in rejecting a command, report an error in DPU_HK indicating the byte-code of the command being rejected in the LSB of the ermo as described in 03691-DPUSDP-01.

Low Level Element Text

Error Collection and Reporting The DPU-CCM CSC provides a centralized error reporting interface, ccmErrEnq(), that other FSW tasks use to report errors. Each time it wakes, ccmTask() checks to see if it is time to form an error/event packet for transmission to the ground. If so, ccmTask() calls ccmHkMkError() to actually create the packet and forward it to DPU-SCUI for transmission to the ground.

FILTER ○ LOCAL ● GLOBAL 0.07

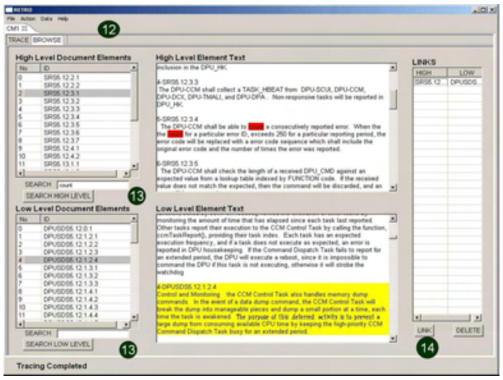
Display The Low Level Elements
● By Requirement
○ By Weigths
○ In Document Order

Fig. 2 RETRO User Interface and Features

developed with a single major use case in mind. This use case involves an IV&V analyst tasked to trace a pair of documents from scratch. One of the current development branches of RETRO deals with additional use cases involving assessment of existing RTMs.

Table 1 Retro Features

ID	Feature
1	Tracing entire dataset at once
2	Tracing elements one at a time
3	Filtering toolbar
4	Filtering option to show top number of links
5	Killing the links that are hidden by the filter
6	Global filtering of candidate link lists
7	Local filtering of candidate link lists
8	View of low-level elements one at a time
9	View of low-level elements in document order
10	View of low-level elements in order of similarity
11	"Freezing" of high-level element tracing
12	Assignment of "Link" and "Not a link" to links
13	Feedback loop
14	Browse tab (Manual Tracing mode)
15	Text search in browse tab
16	Adding links to the RTM from the browse tab



High Level Document Elements

No	ID
0	SRS5.12.2.1
1	SRS5.12.2.2
2	SRS5.12.3.1
3	SRS5.12.3.2
4	SRS5.12.3.3
5	SRS5.12.3.4
6	SRS5.12.3.5
7	SRS5.12.3.6
8	SRS5.12.3.7
9	SRS5.12.4.1
10	SRS5.12.4.2
11	SRS5.13.1.1

SEARCH count

High Level Element Text

inclusion in the DPU_HK.

4-SRS5.12.3.3
The DPU-CCM shall collect a TASK_HBEAT from DPU-SCUI, DPU-CCM, DPU-DCX, DPU-TMALI, and DPU-DPA. . Non-responsive tasks will be reported in DPU_HK.

5-SRS5.12.3.4
The DPU-CCM shall be able to count a consecutively reported error. When the the count for a particular error ID, exceeds 250 for a particular reporting period, the error code will be replaced with a error code sequence which shall include the original error code and the number of times the error was reported.

6-SRS5.12.3.5
The DPU-CCM shall check the length of a received DPU_CMD against an expected value from a lookup table indexed by FUNCTION code. If the received value does not match the expected, then the command will be discarded, and an

Low Level Document Elements

No	ID
0	DPUSDS5.12.0.1
1	DPUSDS5.12.1.2.1
2	DPUSDS5.12.1.2.2
3	DPUSDS5.12.1.2.3
4	DPUSDS5.12.1.2.4
5	DPUSDS5.12.1.3.1
6	DPUSDS5.12.1.3.2
7	DPUSDS5.12.1.3.3
8	DPUSDS5.12.1.4.1
9	DPUSDS5.12.1.4.2
10	DPUSDS5.12.1.4.3
11	DPUSDS5.12.1.4.4

SEARCH

Low Level Element Text

monitoring the amount of time that has elapsed since each task last reported. Other tasks report their execution to the CCM Control Task by calling the function, ccmTaskReport(), providing their task index. Each task has an expected execution frequency, and if a task does not execute as expected, an error is reported in DPU housekeeping. If the Command Dispatch Task fails to report for an extended period, the DPU will execute a reboot, since it is impossible to command the DPU if this task is not executing, otherwise it will strobe the watchdog

4-DPUSDS5.12.1.2.4
Control and Monitoring the CCM Control Task also handles memory dump commands. In the event of a data dump command, the CCM Control Task will break the dump into manageable pieces and dump a small portion at a time, each time the task is awakened. The purpose of this deferred activity is to prevent a large dump from consuming available CPU time by keeping the high-priority CCM Command Dispatch Task busy for an extended period.

LINKS

HIGH	LOW
SRS5.12	DPUSDS

Fig.3 RETRO User Interface: BROWSE tab

RETRO allows analysts to work on *tracing projects*. The work with RETRO must start with an analyst either creating a new project or loading an existing project. To specify a project, the analyst must indicate to RETRO the location of the documents that need to be traced (our GUI shows them as high- and low-level, but any textual artifact may be traced to any other textual artifact). Optionally, the analyst may choose the IR method that RETRO is to use for tracing (the default is vector space retrieval with *tf-idf* term weighting [7]) and select the feedback processing method (the default is Standard Rochio [7]). RETRO invokes the build component to construct the representations of the high- and low-level elements for the selected IR method, after which it displays the main GUI and lets the analyst conduct the tracing. Fig.2 and Fig.3 depict the GUI for the two tracing modes provided by RETRO:

Automatic Tracing Mode, the default mode of RETRO (Fig.2), is designed to let the analyst work with the results of automated tracing methods, and to provide the feedback on the candidate links produced by the automated methods.

Manual Tracing/Browsing Mode, (Fig.3), provides the analyst with the ability to browse high- and low-level documents for the purpose of discovery of any links not found by the automated tools.

We have explicitly identified 13 features of RETRO available for analyst use when tracing. We list these features in Table 1. In Fig.2 and Fig.3, we indicate the GUI location of access to these features. The features are briefly described below.

Tracing all-at-once/One element at-a-time, Feedback. Two buttons on the main GUI screen, "Trace All" and "Trace Current," provide the interface with the selected (at the project start) IR method for tracing. When pressed for the first time, the IR method is executed, and the results are displayed on the screen. Any subsequent presses of either button results in one round of user feedback processing, followed by the execution of the IR method on the new dataset representation. When "Trace All" is pressed, all high-level elements (except those explicitly "frozen" by the user – see below), are traced/retraced. When "Trace Current" is pressed, only the high-level element currently selected in the list of high-level elements is traced (unless it is "frozen," in which case no action is performed).

Filtering of candidate link lists. The filtering tools allow the analyst to reduce the display of the candidate link lists. The analyst specifies a threshold value and then only those low-level documents with relevance weights greater than the given threshold are displayed. The other way of applying filtering is by entering the number of low-level documents that need to be displayed (for example, the "top 5"). The threshold is controlled by a slider bar that can be moved in increments of 0.01 from 0 to 1. The selected filter can have either global or local effect. When the Global radio button is selected, the current filter value applies to candidate link lists for all high-level elements. When the Local radio button is selected, the current filter value applies only to the candidate link list of the currently selected high-level element.

View of low-level elements. There are three ways in which the text of low-level candidate links can be displayed in the tool. First, the low-level links can be displayed **one element at a time**. In this case, only the text of the currently selected low-level element is displayed. The second option is to display the text of **all candidate links in the order that they appear in the low-level document**. In this case, the currently selected low-level element is highlighted. Finally, the candidate links can also be displayed **in the order of their similarity/relevance value**, i.e., in the order their IDs appear in the candidate link list (low-level element list).

Positive/Negative feedback. The main purpose of the RETRO GUI is to solicit analyst feedback on the candidate link lists suggested by the automated methods. There are two steps to the feedback loop. As mentioned above, the "Trace All" and "Trace Current" buttons serve to start the feedback processing loop. The actual feedback is provided by selecting a low-level element, right-clicking the mouse and selecting one of the three options: "Link," "Not A Link," or "Default." Selection of "Link" constitutes positive feedback: the analyst is explicitly marking the current link as belonging to the final RTM. Selection of "Not A Link" constitutes negative feedback: the analyst explicitly excludes the link from the final RTM. Selection of "Default" means that the analyst is not ready to provide explicit feedback on the current link. All links are marked "Default" when they are first added to the candidate

RTM by the automated methods. The analyst also has an option of changing “Link” and “Not a link” assignments back to “Default.” “Links” are highlighted in green, while elements classified as “Not a link” are highlighted in red.

“Freezing” of high-level elements. Anytime the “Trace All” button is pressed, the automated methods retrace all candidate links. To allow analysts more freedom in how they approach tracing tasks, RETRO allows the analysts to **“freeze”** individual high-level requirements – i.e., ensure that they are not retraced when the “Trace All” button is pressed. This feature may be useful for analysts who prefer to trace element-by- element, rather than iteration-by-iteration. To freeze a candidate link list for a high-level requirement, the analyst needs to select a high-level requirement, right-click the mouse button, and select the “Postpone Analysis” option. The change of high-level element status is reflected in the list of high-level requirements.

Browse tab functionality. The “Trace” tab of RETRO lets the analyst evaluate candidate links returned by the automated methods and fix any discovered errors of commission. However, the “Trace” tab interface is not convenient for searching for errors of omission. The “Browse” tab has been designed specifically to address this shortcoming of the “Trace” tab. The “Browse” tab consists of the lists of high- and low-level element IDs, presented in the respective document orders, and two text windows, displaying the high- and low-level requirements. The analyst can browse both documents, select pairs of high- and low-level requirements and, if errors of omission are discovered, add newly discovered low-level elements to the RTM. The list of discovered errors of omission is shown on the right side of the tab, and the links are added to the candidate link lists in the “Trace” tab, with the status set to “Link.” RETRO also provides a simple text search feature for both high- and low-level documents in the “Trace” tab.

4 Validation

In this section, we present the design of the case study, the results, as well as evaluation of the results.

4.1 Case study design

The case study was conducted with a group of thirty (30) students enrolled in a graduate-level requirements engineering course taught at the University of Kentucky during the Spring 2006 semester. There were two groups: those doing tracing manually, and those using RETRO. Students who had previously performed tracing were identified and put into the manual group (there were four such students). Next, the remaining students were divided until two groups of fifteen (15) students existed.

Each group was then taken to a separate location where they received written instructions and a brief background of the task. Students were also given a list of common acronyms used in the data set to assist with the task. Students were not told anything about the task of the other group. Both groups were assigned the same tracing task: to trace twenty-two (22) high-level requirement elements to fifty-two (52) design elements (a subset of the CM-1 dataset, a NASA scientific instrument[22]). Each group was asked to use a different method. Group 1 was asked to perform the

Table 2 Task Assessment Questions

Number	RETRO Group	MANUAL Group
1	The project was simple to complete	
2	The project could be completed quickly	
3	The project was tedious	
4	RETRO was easy to use	
5	If I were performing a similar task in the future, I would want to use a software tool to assist	If I were performing a similar task in the future, I would want to use a software tool to assist
6	I would rather have completed the project by hand than use RETRO	I would rather have completed the project by hand than use a software tool
7	It probably took less time to use RETRO than it would have to complete the project by hand	It probably would have taken less time to use a software tool to complete the project than it did by hand

Table 3 Comparison of Means - 2 Groups

	Recall	*Precision*	*Total Time (minutes)*
Manual Group (Group 1)	0.33	0.24	120.67
RETRO Group (Group 2)	0.70	0.13	41.88
T-Test (p-value)	0.001	0.01	0.0004

Table 4 Comparison of Means - 3 Groups

	Recall	*Precision*	*Total Time (minutes)*
RETRO Gr, Default=Link (Group 2a)	0.979	0.040	42.5
RETRO Gr, Default=No link (Group 2b)	0.48	0.199	41.25
Manual Gr. (Group 1)	0.330	0.243	120.667
T-Test (Group 2a and 2b)	0.0002	0.014	**0.970**
T-Test (Group 2a and 1)	$9x10^{-09}$	$3x10^{-05}$	0.002
T-Test (Group 2 b and 1)	0.019	**0.301**	0.005

tracing and produce an RTM manually. The members of the other group, Group 2, were given a brief introduction to RETRO and were asked to use it to complete the tracing assignment.

Students in both groups were asked to record the amount of time spent on the task. Group 2 students were asked to record the time spent using the tool, but not to include installation time. Additionally, a post-experiment survey was given to students in both groups. The survey consisted of common questions (to both groups) as well as questions specific to the nature of the process employed by each group. Table 2 contains the list of questions from the survey we have tracked in this study. In all questions, student response was measured on the five-point scale: *"strongly agree"(5), "agree"(4), "no opinion"(3), "disagree"(2), and "strongly disagree"(1).*

In addition, students in Group 2 were asked to identify which of the 14 features of RETRO they used, and report how helpful the features were, also using a five-point scale (5 – very helpful, 1 – annoying).

In the end, 11 students from Group 1 submitted the RTM and survey, and 12 RTM and survey submissions were collected from Group 2 students. Out of these, we eliminated two data points from Group 1 (one student submitted an incomplete task, two other students worked together – we elected to treat their submissions as one). In addition, we encountered differences in the interpretation of the task within Group 2. Links that are not explicitly marked as a link or not a link are shown by RETRO as

"Default." There was some confusion as to whether default entries would be considered to be links (and would become part of the final RTM submitted by the student) or would not be considered links (and would be excluded from the final RTM).

We administered a short one question post-task survey, asking how each student in Group 2 viewed the "Default" links. Analysis showed that some students did not fully understand the task, which lead to disqualification of three submissions. Based on the treatment of "Default" links, Group 2 was split into two sub-groups , which we refer to as Group 2a ("default" links included in the RTM) and Group 2b ("default" links not included in the RTM).

There were four and five students in these sub-groups, respectively. This left us with nine (9) data points in each of the groups. Data from the qualitative survey was compiled and the student RTM submissions were checked against the answer set (the actual RTM) for the data set. We attempted to limit internal validity threats by validating the tools and processes we used for data collection and analysis. Another possible threat to internal validity is that of selection. It is possible that some students had prior experience with tracing and/or with tools such as RETRO that would give them an unfair advantage in the study. We attempted to mitigate this risk by placing all students who said they had prior tracing experience in the manual group (Group 1).

Another possible threat to validity is that students may have felt that they needed to provide positive feedback on the surveys (specifically about the tool). While it was emphasized to both groups that the task had no bearing on their grades, it may still have been uncomfortable for students to criticize work that was known to be related to the research of the professor. A threat to external validity (generalization of results) for our work is the use of graduate students. However, Host et al [26] found that students perform the same as professionals on small tasks of judgment.

4.2 Results

Quantitative Results. The quantitative results (recall, precision, and total time to complete the tracing) are shown in Tables 3 and 4. We note that the CM-1 specification used for this experiment was equally unfamiliar to all students, and contained requirements that were hard for students to trace. We did not

Table 5 RETRO Features Used by Group 2

Student	1	2	4	5	6	7	8	9	10	11	12	13	14
A*	5	4	4	4	5	2	4	0	4	0	3	4	0
B	0	4	0	0	3	4	5	0	4	0	4	0	0
C	5	4	5	4	0	5	5	4	4	5	5	5	5
D	4	4	0	0	5	5	5	0	5	0	5	0	5
E*	5	4	4	4	0	0	0	0	0	0	0	0	0
F*	5	0	4	0	5	5	5	0	4	0	0	0	0
G	4	3	0	0	4	0	0	0	2	0	4	0	0
H	3	3	3	3	3	4	4	3	3	3	3	3	3
I*	5	4	5	4	3	4	4	5	5	4	5	3	2
#used	8	8	6	5	7	7	7	3	8	3	7	4	4
Sum	36	30	25	19	28	29	32	12	31	12	29	15	15
Mean	4 5	3.75	4.16	3.8	4	4.14	4.57	4	3.87	4	4.14	3.75	3.75

* Group 2a students (A.E F.I).

Feature
1. Trace All
2. Trace One
4 Global Filtering
5. Local Filtering
6. Display: One Link
7. Display: Document Order
8. Display: By Similarity
9. Freeze Elements
10. Yes/No Links
11. Feedback
12. Browse tab
13. Text Search in Browse
14. Add links in Browse tab

Table 6 Survey Responses (Group2:RETRO).

Stu dent	*Q#1*	*#2*	*#3*	*#4*	*#5*	*#6*	*#7*
A*	4	4	2	4	4	2	4
B	4	4	3	4	4	3	3
C	5	4	2	4	4	2	4
D	3	2	3	2	3	4	3
E*	3	4	2	2	4	2	4
F*	3	3	3	2	4	2	4
G	4	4	4	4	3	3	3
H	3	2	2	4	4	1	4
I *	4	4	2	4	5	1	4
Mean:	3.67	3.44	2.56	3 33	3 89	2.22	3.67

expect students to produce accurate RTMs. Rather, we wanted to study the process the students used, and whether or not this process bore any effect on the accuracy. Table 3 depicts the results when the Manual (Group 1) and Tool (Group 2) groups were analyzed. Table 4 depicts the results when we consider three groups: Group 1, Group 2a, and Group 2b. In each table, we have shown the means as well as the results of the Student t-test (statistical significance). We ran a two-sided test with samples with equal variance.

As can be seen from Table 3, the students with RETRO (Group 2) built RTMs that had higher recall (found a higher percentage of the correct links) than those without RETRO (70.1% recall versus 33% recall). This result was statistically significant (as were all results in Table 3). The students doing manual tracing built RTMs with much higher precision (24.2% as compared to 12.8%) than those using RETRO. That is, their final RTMs did not contain as many “false positives” as RETRO RTMs. Not surprisingly, it took the Manual group almost three times as long to complete the task (120.66 minutes as compared to 41.8 minutes) as the RETRO group.

Examining Table 4, we can see that the students who used RETRO and assumed that “default” was a link had a much higher recall than any other group, a statistically significant result. Precision was much higher for the RETRO group who believed that “default” was not a link than for those who believed it was a link (19.8% versus 4%) and this was statistically significant. This difference is explained by the fact that many of the default links (counted for Group 2b but not for Group 2a) were false positives, however, default links also captured many true links. There was not a statistically significant difference in precision between Group 2a and Group 1, however (t-test of 0.30). The total time was not statistically different between the two sub-groups using RETRO, but was statistically significant between both RETRO subgroups and the manual group.

Use of RETRO features. The results of our survey on RETRO feature use, conducted for Group 2 students, is shown in Table 5. Each column lists the students’ assessment of usefulness of specific features of RETRO on the 1-5 scale, with 5 being “useful” and 0 meaning that the student reported not using the feature (for convenience, we repeat the feature list from Table 1, sans #3, in the right-hand side of Table 5).

We observe that five students used over half of the tracked features (with the mean number of features used being 8.55, and median being 8), while three students used only 4-5 features. Eight out of nine students used “Trace All” and “Trace Current” features and the assignment of “yes” and “no” links. All but two students visited the “Browse tab,” but only four students tried either text searches or link assignment in that tab.

Relevance feedback loop, perhaps the most powerful feature of RETRO, was tried the least – only three students used it. Finally, we see that students had an overall positive impression of the features they used: no feature was rated lower than 3.75 on average.

Task Assessment. Tables 6 and 7 show student answers to survey questions specified in Table 2. Group 2 students (Table 6) tended to agree with most of the statements presented to them. In particular, students found the assignment relatively simple (3.6), agreed that it could be completed relatively quickly (3.4), agreed that RETRO was reasonably easy to use (3.33), and specified that they would prefer to use a software tool for similar tasks in the future (3.67).

In addition, they were in mild disagreement with the statement that the assignment was tedious (2.56), and stated that they would not have preferred to complete the assignment by hand instead (2.22). Students from Group 1 similarly agreed that the task was relatively simple (3.44). At the same time, Group 1, unlike Group 2, thought that the task could not be completed quickly, and declared it to be rather tedious (3.67).

Table 7 Survey Responses (Group 1:Manual).

Stu dent	*Q#1*	*#2*	*#3*	*#5*	*#6*	*#7*
J	4	2	4	5	2	4
K	4	2	4	4	2	4
L	4	2	2	5	1	5
M	2	2	4	4	2	4
N	4	4	3	4	2	5
O	2	3	4	4	2	4
P	4	2	4	5	2	5
Q	2	2	4	5	3	5
R	5	2	4	5	2	4
Mean:	3.44	2 33	3.67	4.56	2	4.44

Additionally, they all have voiced strong support for the use of a software tool for such projects in the future, and expressed a strong opinion that their task could have been accomplished faster with the use of a software tool.

4.3 Evaluation

The study that we undertook had two components: a quantitative component and a qualitative component. We observed that students who used RETRO and decided for themselves that only links explicitly marked "yes" should be reported produced the most accurate results: better recall than Group 1 students, with similar precision. This approach to tracing with RETRO is the correct one for the default RETRO use case – tracing of artifacts for Verification and Validation purposes.

From the usability standpoint, we observed that whenever students chose to use specific features of RETRO, they, in general, found them useful. We also observed that the majority of students chose to use most of the RETRO features available to them. Perhaps the only negative observation is our relative lack of data about the use of relevance feedback in tracing: this is an issue we are planning to concentrate on in future experiments.

We also observed that users of RETRO, in general, felt much better about the task, and felt much better about their ability to deal with the task than students who had to trace manually. The latter group, on the other hand, expressed very strong feelings about the tedium of the assignment and about their desire to use an automated tool for future tasks.

5 Related Work

Ramesh et al. [10] propose a reference model for requirements tracing. In [10], Ramesh elaborates on the factors influencing requirements traceability practice. Spanoudakis [11] uses heuristic traceability rules to trace textual requirements to object models. Cleland-Huang et al. [12] propose an event-based traceability technique to perform impact analysis on proposed changes. Using a prototype tool, Zisman et al. [13] demonstrate their approach for automatic generation of bidirectional traceability links.

Schneidewind defines maintenance as the process of designing and integrating consistent changes to existing software [14]. Traceable software is implicitly easier to maintain because one can easily see how portions of requirements, design, and code relate through the RTM. Through tracing, one can see how a change introduced during maintenance will affect other code portions. Bubel and Balser describe requirements traceability as a "continual alignment between the stakeholder requirements and system evolution… after each modification" and show how context-based constraints (CoCons) can support automation of this process [15]. Research on methods used to trace artifacts for maintenance purposes has also been completed using Model Driven Architecture where model dependencies are encoded and model relationships help ensure that maintenance changes do not introduce inconsistencies [16].

Just as side-effects analysis [17, 18] is valuable during maintenance to identify the impact of code changes on the execution process, tracing can help identify how changes within one phase will affect artifacts in other phases of the software life cycle. Work has been completed on tracing particular code features in order to benefit the maintenance phase of the software life cycle [19, 20]. De Lucia et al. address the usefulness of requirements tracing tools over discovering related artifacts by hand during maintenance in [21]. Likewise, Greevy and Ducasse apply tracing practices to discover change impact during maintenance [20].

Antoniol et al. [24,25] and Marcus and Maletic [27] have used a variety of traditional IR methods (vector space retrieval and probabilistic IR for Antoniol and Latent Semantic Indexing for Marcus and Maletic) to automate tracing of textual artifacts to code. Their approach is similar to our work on tracing between textual artifcats [1,2,3], which lead to the creation of RETRO.

6 Conclusions and Future Work

As stated in the introduction, the requirements traceability matrix is an important artifact for software maintenance. Unfortunately, it is not often constructed or kept up-to-date. We believe that automated methods for generating RTMs (and hence regenerating RTMs when changes are introduced) can thus help to improve software maintenance. We undertook a study to see if our traceability tool, RETRO, would ease the burden of RTM generation. Further, we wanted to examine the usability of the new version of RETRO.

We found that overall, students using RETRO "correctly" (see Section 4.3.) produced the most accurate results. We also found that the majority of the tracked RETRO features were used by the students and were deemed useful by them. In addition, the surveys showed that the RETRO group liked the tool and felt that it made the task faster. Manual tracers wished that they had a tool and found their task to be tedious and time consuming.

We cannot make broad generalizations of these results as we undertook a small study with a small dataset using graduate students. However, the results do indicate that information retrieval traceability tools, such as RETRO, can assist with RTM generation, which is an important part of software maintenance. Based on this study, items for future work include improving the precision of RETRO methods and simplifying the tracing process.

7 Acknowledgements

This work is sponsored by NASA under grant NNG05GQ58G. Our thanks to Stephanie Ferguson, Marcus Fisher, Wes Deadrick, Ken McGill, and Tim Menzies. We thank the students of CS 617, Spring 2006. We thank Mike Chapman and the Metrics Data Program for access to the CM-1 dataset. We also thank Sarah Howard and James Osborne, who worked on early versions of the software used for the evaluation.

References

[1] Jane Huffman Hayes, Alex Dekhtyar and James Osborne, (2003), Improving Requirements Tracing via Information Retrieval, in *Proc., Intl. Requirements Engineering Conference (RE'2003)*, pp.151–161, September 2003, Monterey, CA.

[2] Jane Huffman Hayes, Alex Dekhtyar, K.S. Sundaram and Sarah Howard, (2004), Helping Analysts Trace Requirements: An Objective Look, in *Proceedings, International Requirements Engineering Conference (RE'2004)*, pp. 249-261, September 2004, Kyoto, Japan.

[3] Jane Huffman Hayes, Alex Dekhtyar and K.S. Sundaram, (2006), Advancing Candidate Link Generation For Requirements Tracing: the Study of Methods, *IEEE Transactions of Software Engineering*, Vol 32, No 1., pp. 4-19, January, 2006.

[4] Suresh Yadla, Jane Huffman Hayes and Alex Dekhtyar, (2005), Tracing Requirements to Defect Reports, *Innovations in Systems and Software Engineering: A NASA Journal.*, Vol. 1, No. 2 (September 2005), pp. 116–124.

[5] Jane Huffman Hayes, Alex Dekhtyar, Senthil Sundaram*, "Text Mining for Software Engineering: How Analyst Feedback Impacts Final Results," *Proceedings of Workshop on Mining of Software Repositories (MSR)*, associated with ICSE 2005, St. Louis, MO, May 2005, pp. 58 - 62.

[6] Jane Huffman Hayes, Alex Dekhtyar, and Senthil Sundaram, (2006), Advances in Dynamic Generation of Traceability Links: Two Steps Closer to Full Automation?, *University of Kentucky Technical Report* TR 451-06, Februrary 2006.

[7] Baeza-Yates, R. and Ribeiro-Neto, B. Modern Information Retrieval, *Addison-Wesley*, 1999.

[8] Jane Huffman Hayes, Alex Dekhtyar and K.S. Sundaram, (2005), Improving After the Fact Tracing and Mapping to Support Software Quality Predictions, *IEEE Software*, Vol. 22, No. 6 (November/December), pp. 30-37.

[9] Ken McGill, Wes Deadrick, Jane Hayes, Alex Dekhtyar, (2006), Houston, We Have a Success Story: Technology Transfer at the NASA IV&V Facility, in *Proceedings, International Workshop on Technology Transfer in Software Engineering* (WOTTSE'2006), Shaghai, China, May 2006.

[10] Ramesh, B., Factors Influencing Requirements Traceability Practice, *Communications of the ACM,* December 1998, Volume 41, No. 12, pp. 37-44.

[11] Spanoudakis, G. Plausible and Adaptive Requirement Traceability Structures, in *Proc. 14th International Conference on Software Engineering and Knowledge Engineering*, pp. 135–142, Ischia, Italy, July 2002.

[12] Cleland-Huang, J., Chang, C.K., Sethi, G., Javvaji, K.; Hu, H., Xia, J., Automating speculative queries through event-based requirements traceability, in *Proceedings of the IEEE Joint International Requirements Engineering Conference (RE'02)*, Essex, Germany, 9-13 September, 2002, pages: 289- 296.

[13] A. Zisman, G. Spanoudakis, E. Pérez-Miñana, and P. Krause. Tracing Software Requirements Artefacts, *in Proc. 2003 International Conference on Software Engineering Research and Practice (SERP 2003)*, Las Vegas, June 2003

[14] Schneidewind, NF: The State of Software Maintenance. *IEEE Transactions on Software Engineering*, 13(1987)3, pp. 303-310.

[15] Felix Bubel, Michael Balser, Tracing Cross-Cutting Requirements via Context-Based Constraints, in *Proc. Ninth European Conference on Software Maintenance and Reengineering (CSMR'05)*, pp. 80-90, 2005.

[16] Igor Ivkovic, Kostas Kontogiannis, Tracing Evolution Changes of Software Artifacts through Model Synchronization, in *Proc. 20th IEEE International Conference on Software Maintenance (ICSM'04)*, pp. 252-261, 2004.

[17] Atanas Rountev, Precise Identification of Side-Effect-Free Methods in Java, in *Proc. 20th IEEE International Conference on Software Maintenance (ICSM'04),* pp. 82-91, 2004.

[18] B. G. Ryder,W. Landi, P. Stocks, S. Zhang, and R. Altucher. A schema for interprocedural modification side-effect analysis with pointer aliasing, *ACM Trans. Programming Languages and Systems*, 23(2):105–186, Mar. 2001.

[19] Andrew David Eisenberg, Kris De Volder. Dynamic Feature Traces: Finding Features in Unfamiliar Code, in *Proc. International Conference on Software Maintenance, (ICSM'05),* pp. 337-346, 2005.

[20] Orla Greevy, Stephane Ducasse and Tudor Girba Analyzing Feature Traces to Incorporate the Semantics of Change in Software Evolution Analysis, in *Proc. International Conference on Software Maintenance (ICSM'05)*, 2005.

[21] De Lucia A., Fasano F., Francese R., and Oliveto R., Recovering Traceability Links between Requirement Artefacts: a Case Study, in *Proc. 16th International Conference on Software Engineering and Knowledge Engineering - Workshop on Knowledge Oriented Maintenance,* Banff, Alberta, Canada, Knowledge Systems Institute, USA, pp. 453-456, 2004. [22] MDP Website, CM-1 Project, http://mdp.ivv.nasa.gov/mdp_glossary.html#CM1.

[23] S. R. Schach, B. Jin, L. Yu, G. Z. Heller and J. Offutt; Determining the distribution of maintenance categories: survey versus management, *Empirical Software Engineering,* 8, December 2003, pp. 351-366.

[24] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering Traceability Links

between Code and Documentation. *IEEE Transactions on Software Engineering*, 28(10):970-983, 2002.

[25] G. Antoniol, B. Caprile, A. Potrich, and P. Tonella. Design-Code Traceability for Object Oriented Systems. *Annals of Software Engineering*, 9:35--58, 1999.

[26] Høst M, Regnell B, Wohlin C. Using students as subjects – A comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, 5(3):210–214, 2000.

[27] A. Marcus and J. Maletic, Recovering Documentation-to-Source Code Traceability Links using Latent Semantic Indexing. *In Proceedings of the Twenty-Fifth International Conference on Software Engineering (ICSE)*, 2003, pp. 125-135, 2003.