Mining Requirements Traces : A Decade Long Survey

George Mathew
Department of Computer Science
North Carolina State University
Raliegh, NC, USA
george2@ncsu.edu

ABSTRACT

Requirements traceability is a critical software engineering activity that provides support for numerous tasks such as impact analysis, compliance verification and coverage analysis. Automated trace retrieval methods came into prominence since early 2000s and significantly reduce the cost and effort needed to create and maintain requirements traces. These techniques generates low precision and is augmented with manual evaluation with business analysts. In this report, a study is performed to review various techniques used in the last decade to mine traces. These techniques are mostly automated with some of them using manual augmentation to filter the queries. The report also describes the datasets used to evaluate these techniques and suggests what the road head in requirements traceability for the next decade.

Keywords

Requirements; Information Search; Requirements Traceability; Text Mining

1. INTRODUCTION

Requirements traceability is defined as the ability to track the life of a requirement back to its source documents and forward to the downstream work products in which it is realized [**?**]. Traceability links are generally created and maintained by project stakeholders using spreadsheets, databases, or specialized trace features of commercial requirements management tools. However, in nontrivial projects, the number of traceability links can grow very large, and as a result the manual effort required to establish and maintain such traces is often inhibitive.

Various different approaches were used for information retrieval in dynamically generating traceability links. Most of the popular approaches have utilized Vector Space Modelling (VSM) [**?**], probabilistic approaches [**?**] or Latent Semantic Indexing(LSI) [**?**, **?**]. Prior to these techniques, standard software engineering practices of requirements traceablity were used which refers to "the ability to track a requirement from its origins back to its rationale and downstream to various work products that implement the requirement is software"[**?**]. To trace the requirements, software developers are often forced to manually pore over documentation manuals to identify relevant sections and then painstakingly trace them to product level requirements or implemented code. These traces are typically represented in trace matrices.

In this paper, we discuss various techniques for automatically obtaining requirement traces. Section **??** starts of with retrieving traceability links between code and documentation. Section **??** discusses utilizing support evidence for improving dynamic requirements traceability. Sections **??**, **??** and **??** discusses utilizing wikipedia, machine learning techniques and search engines to improve user's query intents. Further, sections **??**, **??** and **??** highlights enhancement of candidate links, reformulating queries to yield better results and techniques for more intelligent trace retrievals

2. REQUIREMENTS TRACEABILITY

2.1 Code & Documentation Traceability

Code and Document traceability was explored by *Antoniol et.al* [**?**] in their 2002 publication in IEEE Transactions of Software Engineering.

Traceability links play a vital role in acquiring data for requirements. Links between areas of code and related sections of free text documents, such as an application domain handbook, a specification document, a set of design documents, or manual pages, aid both top-down and bottom-up comprehension. The links between code and other sources of information are a sensible help to perform the combined analysis of heterogeneous information and, ultimately, to associate domain concepts with code fragments. Traceability links between the requirement specification document and the code are a key to locate the areas of code that contribute to implement specific user functionality. This helps assess the completeness of an implementation with respect to stated requirements, to devise complete and comprehensive test cases, and to infer requirement coverage from structure coverage during testing. Traceability links between requirements and code can also help to identify the code areas directly affected by a maintenance request as stated by an end user. Tracing code to free text documents are a sensible help to locate reused candidate components. Traceability Links are retrieved as shown in Figure **??**.

In their work, recovering traceability links between free

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

text documentation and source-code components cannot be simply based on compiler techniques because of the difficulty of applying syntactic analysis to natural language sentences. Hence the premise of their work is that programmers use meaningful names for program items, such as functions, variables, types, classes, and methods. The analysis of mnemonics can help to associate high-level concepts with program concepts, and vice-versa. The names of program items are used as a clue to suggest concepts implemented in the code. After this preprocessing they, propose two different methods, a **Probabilistic Information Retrieval Model** and **Vector Space Information Retrieval Model**.

2.2 Dynamic Requirements Traceability

2.3 Wikipedia for user's query intent

2.4 Machine Learning Techniques

2.5 Search Engines Augmentation

2.6 Enhancing Candidate Links

2.7 Query Reformulation

2.8 Intelligent Trace Retrieval

Typically, the body of a paper is organized into a hierarchical structure, with numbered or unnumbered headings for sections, subsections, sub-subsections, and even smaller sections. The command `\section` that precedes this paragraph is part of such a hierarchy.[1] LaTeX handles the numbering and placement of these headings for you, when you use the appropriate heading commands around the titles of the headings. If you want a sub-subsection or smaller part to be unnumbered in your output, simply append an asterisk to the command name. Examples of both numbered and unnumbered headings will appear throughout the balance of this sample document.

Because the entire article is contained in the **document** environment, you can indicate the start of a new paragraph with a blank line in your input file; that is why this sentence forms a separate paragraph.

2.9 Type Changes and *Special* Characters

We have already seen several typeface changes in this sample. You can indicate italicized words or phrases in your text with the command `\textit`; emboldening with the command `\textbf` and typewriter-style (for instance, for computer code) with `\texttt`. But remember, you do not have to indicate typestyle changes when such changes are part of the *structural* elements of your article; for instance, the heading of this subsection will be in a sans serif[2] typeface, but that is handled by the document class file. Take care with the use of[3] the curly braces in typeface changes; they mark the beginning and end of the text that is to be in the different typeface.

[1]This is the second footnote. It starts a series of three footnotes that add nothing informational, but just give an idea of how footnotes work and look. It is a wordy one, just so you see how a longish one plays out.

[2]A third footnote, here. Let's make this a rather short one to see how it looks.

[3]A fourth, and last, footnote.

You can use whatever symbols, accented characters, or non-English characters you need anywhere in your document; you can find a complete list of what is available in the *LaTeX User's Guide*[?].

2.10 Math Equations

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

2.10.1 Inline (In-text) Equations

A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin. . .\end` construction or with the short form `$. . .$`. You can use any of the symbols and structures, from α to ω, available in LaTeX[?]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\to\infty} x = 0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

2.10.2 Display Equations

A numbered display equation – one set off by vertical space from the text and centered horizontally – is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\to\infty} x = 0 \tag{1}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty} x_i = \int_0^{\pi+2} f \tag{2}$$

just to demonstrate LaTeX's able handling of numbering.

2.11 Citations

Citations to articles [?, ?, ?, ?], conference proceedings [?] or books [?, ?] listed in the Bibliography section of your article will occur throughout the text of your article. You should use BibTeX to automatically produce this bibliography; you simply need to insert one of several citation commands with a key of the item cited in the proper location in the `.tex` file [?]. The key is a short reference you invent to uniquely identify each work; in this sample document, the key is the first author's surname and a word from the title. This identifying key is included with each item in the `.bib` file for your article.

The details of the construction of the `.bib` file are beyond the scope of this sample document, but more information can be found in the *Author's Guide*, and exhaustive details in the *LaTeX User's Guide*[?].

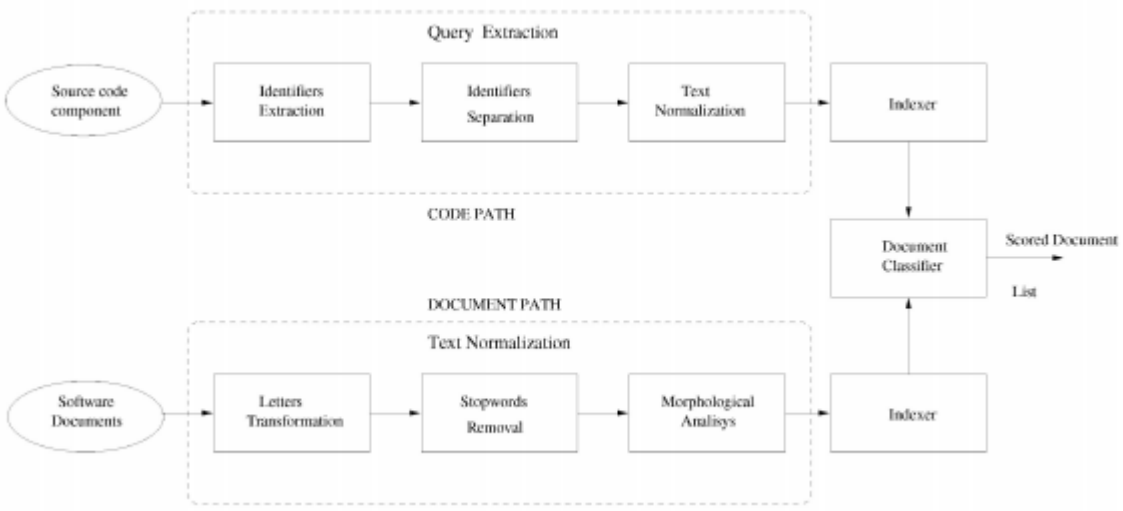

Figure 1: A sample black and white graphic that needs to span two columns of text.

Table 1: Frequency of Special Characters

Non-English or Math	Frequency	Comments
$\emptyset$	1 in 1,000	For Swedish names
π	1 in 5	Common in math
\$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

This article shows only the plainest form of the citation command, using `\cite`. This is what is stipulated in the SIGS style specifications. No other citation format is endorsed or supported.

2.12 Tables

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material is found in the *LaTeX User's Guide*.

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed dvi output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed dvi output of this document.

2.13 Figures

Figure 2: A sample black and white graphic.

Figure 3: A sample black and white graphic that has been resized with the `includegraphics` command.

Like tables, figures cannot be split across pages; the best placement for them is typically the top or the bottom of the page nearest their initial cite. To ensure this proper "floating" placement of figures, use the environment **figure** to enclose the figure and its caption.

This sample document contains examples of **.eps** files to be displayable with LaTeX. If you work with pdfLaTeX, use files in the **.pdf** format. Note that most modern TeX system will convert **.eps** to **.pdf** for you on the fly. More details on each of these is found in the *Author's Guide*.

As was the case with tables, you may want a figure that spans two columns. To do this, and still to ensure proper "floating" placement of tables, use the environment **figure*** to enclose the figure and its caption. and don't forget to end the environment with figure*, not figure!

2.14 Theorem-like Constructs

Other common constructs that may occur in your article are the forms for logical constructs like theorems, axioms, corollaries and proofs. There are two forms, one produced by the command `\newtheorem` and the other by the command `\newdef`; perhaps the clearest and easiest way to distinguish them is to compare the two in the output of this sample document:

This uses the **theorem** environment, created by the `\newtheorem` command:

THEOREM 1. *Let f be continuous on $[a, b]$. If G is an antiderivative for f on $[a, b]$, then*

$$\int_a^b f(t)dt = G(b) - G(a).$$

The other uses the **definition** environment, created by

Table 2: Some Typical Commands

Command	A Number	Comments
`\alignauthor`	100	Author alignment
`\numberofauthors`	200	Author enumeration
`\table`	300	For tables
`\table*`	400	For wider tables

Figure 4: A sample black and white graphic that needs to span two columns of text.

Figure 5: A sample black and white graphic that has been resized with the `includegraphics` command.

the `\newdef` command:

Definition 1. If z is irrational, then by e^z we mean the unique number which has logarithm z:

$$\log e^z = z$$

Two lists of constructs that use one of these forms is given in the *Author's Guidelines*.

There is one other similar construct environment, which is already set up for you; i.e. you must *not* use a `\newdef` command to create it: the **proof** environment. Here is a example of its use:

Proof. Suppose on the contrary there exists a real number L such that

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = L.$$

Then

$$l = \lim_{x\to c} f(x) = \lim_{x\to c} \left[gx \cdot \frac{f(x)}{g(x)} \right] = \lim_{x\to c} g(x) \cdot \lim_{x\to c} \frac{f(x)}{g(x)} = 0 \cdot L = 0,$$

which contradicts our assumption that $l \neq 0$. □

Complete rules about using these environments and using the two different creation commands are in the *Author's Guide*; please consult it for more detailed instructions. If you need to use another construct, not listed therein, which you want to have the same formatting as the Theorem or the Definition[**?**] shown above, use the `\newtheorem` or the `\newdef` command, respectively, to create it.

A *Caveat* for the TEX Expert

Because you have just been given permission to use the `\newdef` command to create a new form, you might think you can use TEX's `\def` to create a new command: *Please refrain from doing this!* Remember that your LATEX source code is primarily intended to create camera-ready copy, but may be converted to other forms – e.g. HTML. If you inadvertently omit some or all of the `\def`s recompilation will be, to say the least, problematic.

3. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LATEX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

4. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.