Literature Survey of Requirements Traceability

Brent Seidel
IEE 598

December 10, 2010

Contents

List of Figures

Chapter 1

Introduction

“Requirements traceability is defined as the ability to describe and follow the life of a requirement in both directions, towards its origin or towards its implementation, passing through all the related specifications.”[IEE02]

Refer to Figure 1.1 and consider, for example, designing a new commercial airplane. To be most useful to customers, the airplane needs to be able to use short runways (DCO). After the minimum runway length is specified, the FAA requirements for the “Accelerate-Stop” distance, defined in 14 CFR 25.109, are used to figure out how quickly the airplane needs to be able to stop (Regulation, Physical Limitations). This, then is a new requirement that flows down to the landing gear to specify how strong it needs to be and to the brakes and tires. Being able to trace between these requirements enables one to determine the impact of one requirement changing. For example, if it was determined that no tires were available that met the original requirements, the tire requirement might be changed. Then the traceability would be used to determine which other requirements were also impacted and they could be evaluated.

Depending on the size of the program, the model in Figure 1.1 may be modified. Small projects may have only one layer of requirements. Large projects may have more. The model may be split over jurisdictional boundaries. For example, a prime contractor may have responsibility for the first layer of requirements, but use sub-contractors to further develop the lower level requirements and the implementation.

In some cases, having requirements traceability is viewed as a measure of system quality and may be mandated by the governing standards[Ram98]. Unfortunately, the implementation of traceability is often haphazard[RJ01]. The problem is, in my experience, that while most people see traceability as a valuable thing, nobody wants to do it themselves.

[IEE96] and [GF94] describe two types of traceability identified as “pre-traceability” and “post-traceability”. Post-traceability is what is typically though of as requirements traceability and enables tracing from the requirements to the design and implementation. Pre-traceability is used to trace the requirements to their origins. Referring back to Figure 1.1, “pre-traceability” is what is above the dotted line, and primarily the DCOs. “Post-traceability” is what is below the dotted line.

[Ram98] and [RJ01] divide organizations into “low-end” and “high-end” users of requirements traceability. The low-end users are typically new to traceability and view it simply as something that has been levied on them from outside. In contrast, the high-end users are experienced with

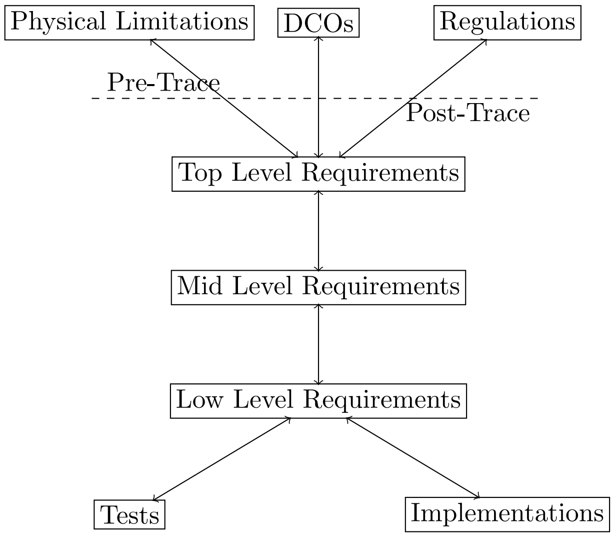

Figure 1.1: Levels of Traceability

traceability and are extracting benefits from it. There should probably be an additional category on non-users of traceability.

In order to improve the quality of traceability, some tools have been developed to attempt to automate the process [IEE05]. These include PRO-ART, described in [IEE96] for tracking "pre-traceability", automatic trace analyzers described in [EG02] and [IEE05] which seek to analyze the requirements and automatically generate the trace information, and [IBM08], a commercial requirements management product which includes traceability information.

Chapter 2

Taxonomy

The taxonomy of requirements traceability is shown in Figure 2.1.

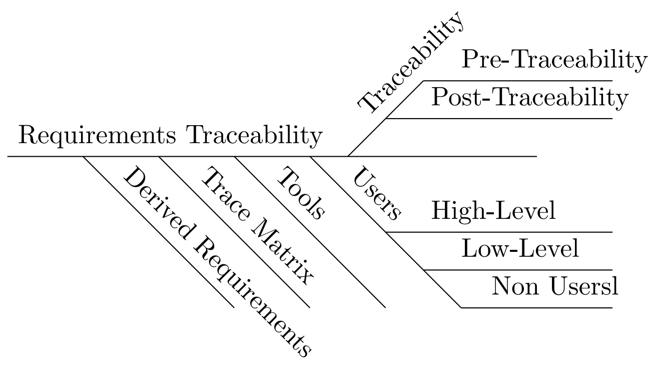

Figure 2.1: Requirements Traceability Taxonomy

2.1 Requirements Traceability

"Requirements traceability is defined as the ability to describe and follow the life of a requirement in both directions, towards its origin or towards its implementation, passing through all the related specifications."[IEE02] This definition can be extended to include the mechanisms of defining and maintaining relationships between artifacts [EG02].

There is broad agreement that having requirements traceability is a good thing [IEE05] (or most any other reference). Indeed, [IEE96] states that it is necessary for building high quality software systems.

2.2 Trace Matrix

The *Trace Matrix*[1] is the artifact that contains the linkages between higher and lower level requirements or between requirements and implementation. The actual implementation is not particularly important. It may be a relational database, a spreadsheet, or other method. The deliverable may be a document or other method. At a minimum, for the low-level traceability users, the trace matrix contains a pointer to the requirement and the implementation. For the high-level traceability users, the *Trace Matrix* will contain additional information.

2.3 Pre-Traceability

This refers to trace from the requirements to the origin of the requirements [IEE96]. Broadly speaking, this would be the DCOs, regulatory requirements, and physical limitations. The difficulty with this is that this information is usually not neatly documented and structured. Much of it is informal, such as meeting minutes or interview notes, which makes it difficult to do manually and impossible to do in a fully automated fashion. PRO-ART, as described in [IEE96] is an attempt to partially automate this process.

2.4 Post-Traceability

This refers to trace between the requirements and the design and implementation [IEE96]. This is what is commonly though of as requirements traceability.

2.5 Low-Level Users

The low level users of traceability view the *Trace Matrix* as a deliverable mandated by the customer or by standards [RJ01]. The *Trace Matrix* generated by low level users typically only provides a link from the requirements to the implementation and test. This level of traceability can be used to generate tests to verify that all of the requirements have been implemented. This trace includes only the post-traceability described above. Of the 26 organizations in the study in [RJ01], nine of them fall into the low level user group.

2.6 High-Level Users

The high level users view the traceability process as an opportunity for satisfying the customer and creating knowledge [RJ01]. In addition to providing simple trace linkages like the low level users' *Trace Matrix*, the high level users add additional information describing the relationship between the items being linked. Also, the high level users include trace links to the customer's "Critical Success Factors". This is essentially including the pre-traceability as well as post-traceability described earlier. Of the 26 organizations in the study in [RJ01], 17 of them fall into the high level user group.

[1]Throughout the rest of this paper, I will be using *Trace Matrix* to refer to the physical artifact of requirements traceability when the implementation doesn't matter. It may be implemented in a database maintained by automatic tools or a sheet of paper maintained by hand.

2.7 Non Users

[RJ01] and [Ram98] cover a description of “High-Level” and “Low-Level” users of traceability. However, they omit a discussion of non-users of traceability. Even among users of traceability, ad hoc tools developed for internal use may not use requirements traceability.

2.8 Tools

[IEE05] notes that manually constructing and maintaining a *Trace Matrix* is costly and often perceived to have costs that outweigh the benefits. This would lead to the group described in Section 2.7.

2.9 Derived Requirements

Derived requirements come from higher level requirements and assumptions. They are typically created either to add implementation details or to fill in holes in the higher level requirements. They can be especially problematic for low-level traceability users since their trace matrix contains no information about the origin of these requirements [RJ01].

Chapter 3

Tools

Traceability is difficult and tedious. Many tools have been developed to help this process. The tools fall into two categories; academic and commercial. Some have already been mentioned. There are several things that a trace tool can help with. Some are easier to implement than others.

However, tools can only go so far. Building and maintaining a *Trace Matrix* is still largely a manual process[EG05].

3.1 Trace Matrix Management

At the most basic level, some tool is needed to manage the *Trace Matrix* itself. Without this, there is no traceability. This can be as simple a sheet of paper with a couple of columns or it can be integrated into a requirements management system. Often they are implemented with a spreadsheet or relational database. PRO-ART[IEE96] is an example of a tool that manages a *Trace Matrix*. In this case, it is focused on pre-traceability.

3.2 Simple Structural Correctness

This is the easy part. Once you have a trace matrix and a list of requirements, it is fairly straightforward to compare the two lists. From this, you can identify a list of requirements that have been missed in the trace matrix and a list of entries in the trace matrix that do not point to valid requirements.

In some cases, if trace tags are embedded in the text along with the requirement, duplicate trace tags may also exist. Should this happen, then entries in the *Trace Matrix* may be ambiguous.

These issues are fairly simple to detect (I've written tools to do this myself). Depending on the sophistication of a requirements management system, they may even be prevented from happening.

3.3 Missing Trace and Incorrect Trace

3.3.1 Testing Based Approach

One approach to determining which requirements trace to which pieces of code is described in [EG02] and [EG05]. This approach requires that tests or "scenarios" be written to test the various requirements. The software is then observed to determine which lines of code are executed for each test.

This approach requires that properly traced tests exist. While it can be useful in tracing between requirements and code, it may be of less value when tracing between levels of requirements.

3.3.2 Keyword Matching Probabilistic Approach

Another approach, described in [IEE05], is to try and identify trace links through matching similar keywords. This is a probabilistic approach and requires a human to identify the true links out of a set of candidate links. As a probabilistic approach, there are two possible problems. One is identifying extra potential trace links. The other is missing true trace links. Since a human is already in the loop, the first problem is not serious until the number of extra links gets large. The second problem is more serious and indicates that all of the requirements need to be searched to find the missing links. As noted in [IEE05], it is best to err on the side of providing to many potential links.

This approach divides potential links into three groups. The first has a high confidence that they are true links and are presented to the user for evaluation. The second has low confidence. The third group has high confidence that these links are not true links and should be rejected. Some strategies are described to attempt to move links from the second group to either the first or third group.

3.3.3 Ontologically Driven Approach

This approach, described in [ASP09], attempts to construct an ontology of the requirements and the software and then match them. The ontologies are represented using first-order logic. The matching returns one of five different relations between the requirements and software. The first is equivalence which represents a total and complete match. The second is a more or less general match. The third is a mismatch which indicates that the particular software does not match the particular requirement. The fourth is overlapping which indicates an overlap between the requirements and software. Finally, if the tool cannot determine any other relationship, it returns "idk" (I don't know).

3.3.4 Evolving the Trace Matrix

Once the *Trace Matrix* has been created (in some fashion), it needs to be maintained. A method of maintaining the *Trace Matrix* is described in [HKL10]. It works by examining changes to the artifacts (requirements, software, etc) and using the current *Trace Matrix* to identify potential impacts.

3.4 Available Tools

The INCOSE website has a page with a list of requirements management tools[1]. One of these tools, DOORs was arbitrarily picked as representative of commercial tools. In addition, an open source tool was also reviewed.

3.4.1 DOORs

DOORs does provide some traceability support[IBM08], however it requires manual entry of the trace links. Once the links are entered, the *Trace Matrix* can be followed up and down. This is adequate for the low-level traceability users[RJ01], but not for high-level users. The additional information would need to be captured separately.

The requirements entered into DOORs can start with the DCOs and be elaborated into lower level requirements. This will support both pre and post traceability[IEE96].

3.4.2 Open Source Requirements Management Tools / Nimble

There are some good reviews for Open Source Requirements Management Tools (OSRMT)[2], but development seems to have stalled. A new project has started called Nimble[3] which aims to be a replacement for OSRMT. Nimble is currently under active development, but has not yet made a beta release. This may be a good project for someone interested in requirements traceability to join.

[1] `http://www.incose.org/productspubs/products/rmsurvey.aspx`
[2] `http://sourceforge.net/projects/osrmt/`
[3] `http://sourceforge.net/projects/nimble/`

Chapter 4

Conclusions

4.1 The Good

Having a complete *Trace Matrix* in place enables one to quickly determine the impact of requirements changes and determine if all of the requirements have been implemented and tested. Conversely, if implementing a requirement proves to be infeasible, one can follow the traceability upwards to determine what other requirements may be impacted. If pre-traceability has been done, this can even lead up to the DCOs and enable negotiations about how badly the customer wants the feature.

4.2 The Bad

Traceability is tedious hard work that nobody wants to do. As a result, the implementation is often haphazard[EG05] and seen as a cost rather than a benefit. It is reported[RJ01] that 4% of the U.S. Defense Department's IT costs are spent on traceability.

4.3 The Ugly

While tools do exist that automate some aspects of traceability, there are many challenges and the problem is far from solved[ASP09]. Once the *Trace Matrix* is created, keeping it maintained is also a difficult problem that is far from solved[HKL10].

Appendix A

Searches Used

For the introductory material, the Google Scholar search engine was used. The search term used was "requirements traceability". This produces the URL `http://scholar.google.com/scholar?q=requirements+traceability&hl=en&btnG=Search&as_sdt=801&as_sdtp=on`

For additional material about, the ASU library website was used.

Bibliography

[ASP09] N. Assawamekin, T. Sunetnanta, and C. Pluempitiwiriyawej. Mupret: An ontology-driven traceability tool for multiperspective requirements artifacts. In *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on*, pages 943 – 948, June 2009.

[EG02] A. Egyed and P. Grünbacher. Automating requirements traceability: Beyond the record replay paradigm. In *Automated Software Engineering, 2002. Proceedings. ASE 2002. 17th IEEE International Conference on*, pages 163 – 171, April 2002.

[EG05] Alexander Egyed and Paul Grünbacher. Supporting software understanding with automated requirements traceability. *International Journal of Software Engineering*, 15(5), October 2005.

[GF94] Orlena C. Z. Gotel and Anthony C. W. Finkelstein. An analysis of the requirements traceability problem. In *Proceedings of the First International Conference on Requirements Engineering*. IEEE Computer Society, 1994.

[HKL10] Youngki Hong, Minho Kim, and Sang-Woong Lee. Requirements management tool with evolving traceability for heterogeneous artifacts in the entire life cycle. In *Software Engineering Research, Management and Applications (SERA), 2010 Eighth ACIS International Conference on*, pages 248 –255, May 2010.

[IBM08] IBM. *Telelogic DOORS Getting Started with Telelogic DOORS*. IBM, version 9.1 edition, 2008.

[IEE96] IEEE International Conference on Requirements Engineering. *PRO-ART: Enabling Requirements Pre-Traceability*, April 1996.

[IEE02] IEEE International Conference on Automated Software Engineering (ASE). *A Framework for Requirements Traceability in UML-based Projects*, September 2002.

[IEE05] IEEE International Conference on Requirements Engineering. *Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability*, number 13th. IEEE Computer Society, 2005.

[Ram98] Balasubramaniam Ramesh. Factors influencing requirements traceability practice. *Communications of the ACM*, 41:37–44, 1998.

[RJ01] Bala Ramesh and Matthias Jarke. Towards reference models for requirements traceability. *IEEE Transactions on Software Engineering*, 2001.