

Just Enough Requirements Traceability

Jane Cleland-Huang
Center for Requirements Engineering
School of Computer Science, Telecommunications and Information Systems
DePaul University
jhuang@cs.depaul.edu

Abstract

Even though traceability is legally required in most safety critical software applications and is a recognized component of many software process improvement initiatives, organizations continue to struggle to implement it in a cost-effective manner. This panel addresses the problems and challenges of requirements traceability and asks questions such as “How much traceability is enough?” and “What kinds of traceability provide cost effective solutions?” Traditional, automated, and lean traceability methods are all discussed.

1. Introduction

Requirements traceability helps developers to control and manage the development and evolution of a software system. It has been defined as the “ability to follow the life of a requirement in both a forward and backward direction” in order to understand the origins of the requirement and also to determine how a requirement has been realized in downstream work products such as design, code, and test cases [1]. In many critical systems, developers are legally required to show not only that each requirement is fully implemented, but also that no additional and untraceable code exists. Commonly accepted standards such as CMMI Level 3, require basic traceability practices to be in place.

Unfortunately the actual task of building a requirements trace matrix (RTM) is time consuming, arduous and error prone. Therefore, despite its usefulness, many organizations fail to implement effective traceability practices which they perceive to be too costly in relation to its benefits. In a seminal study conducted by Gotel and Finkelstein [1], it was found that traceability problems primarily occurred as a result of breakdowns in communication between developers who were hard-pressed for time. Their efforts were further hindered by lack of tool support, and a general perception that the effort required to maintain a requirements traceability matrix (RTM) was excessive in respect to its benefits.

Ramesh showed that many organizations, who did

invest in traceability processes, were what he called “low end users” [2]. These users tended to create basic links between different types of artifacts but failed to benefit from higher levels of automation that would have increased the benefits of their traceability efforts.

This panel addresses the critical question of how an organization can implement just enough requirements traceability to provide essential support mechanisms for change-related activities such as compliance verification, impact analysis, requirements validation, and regression test case selection. It further addresses the important question of whether cost-effective requirements traceability techniques exist, and whether non-traditional methods such as lean-traceability or automated-traceability could replace traditional traceability practices.

2. Traceability Methods

Current traceability practices typically require developers to maintain an RTM, using a spreadsheet, database, word processing software, or a requirements management tool. Analysts are responsible for manually constructing, maintaining, and analyzing links, and for determining which artifacts they intend to trace and at what level of granularity. Tracing at a low level of granularity supports a much more precise form of traceability but can create an excessive amount of work. In fact, Domges showed that too many low-level traces often results in a “useless tangle” of traceability links that are hard to maintain and even harder to understand [3]. In practice, many RTMs that are carefully constructed during early phases of the software development lifecycle, slowly yet consistently deteriorate and become inaccurate and incomplete as the system evolves.

2.1 Automated Traceability

As a result of these problems, a number of researchers have investigated the use of automated traceability methods using information retrieval methods such as the vector space model, semantic indexing, or probabilistic network models to dynamically generate traces at runtime [4,5]. The effectiveness of automated traceability is measured using the standard metrics of recall and

precision, where recall measures the number of correct links that are retrieved by the tool, and precision measures the number of correct links out of the total number of retrieved links. Numerous experiments, conducted using both experimental data sets as well as industry and government data sources, have consistently shown that when recall levels of 90-95% are targeted precision of 10-35% is generally obtainable. This means that automated traceability methods require a human analyst to manually evaluate the candidate links returned by the tool and to filter out the incorrect ones. Automated trace retrieval, while no silver-bullet, is increasingly recognized by industry as a potential traceability solution. Prototype tools such as Poirot [5] and RETRO [4], are currently being used in industrial pilot studies. The new Center of Excellence in Traceability has been established specifically to address these issues [7].

2.2 Lean Traceability

The problems of traditional traceability methods were highlighted in an online discussion of the agile project management discussion group, where group members discussed the idea of adding “Tests over requirements definitions and traceability” as an addendum to the agile manifesto [8]. Although the consensus was that the agile manifesto should not be subject to change, there was rather strong support for the core idea that traditional traceability practices were generally detrimental.

Appleton identified nine specific problems with traditional traceability practices and labeled them as ‘gripes.’ These gripes included the unnecessary creation of trace artifacts and the almost inevitable failure to accurately maintain them; the focus on upfront activities and comprehensive documentation which meant that the important task of writing code and delivering executable product was delayed and had a negative impact on production performance; creating an illusion that real work is being done while in fact time is being wasted developing the trace matrix; focusing on comprehensive documentation rather than the real deliverable of working software; creation of overhead to the change process itself which actually makes change more difficult to implement. Appleton also stated that traditional traceability methods assumed a waterfall lifecycle model and that they required developers to spend time seeking information rather than actually accomplishing the task of software development. His final gripe was that there was a poor fit between traceability and the agile practice of short cycles supported by close customer collaboration.

Agile traceability methods therefore advocate a much leaner approach to traceability based on developers innate knowledge of the system, and use of existing agile artifacts such as test cases to help developers understand how a change in a requirement impacts the source code.

3. Open Questions

This panel will address many of the open questions raised in this ongoing debate on traceability. As traceability is legally required in many safety critical software systems, the question is “what is the right amount of traceability?” and “what kind of traceability can be used to achieve the desired results in a cost-effective way?” Organizations trying to improve their ability to manage change effectively must ask very similar questions. The panelists, who are drawn from a wide variety of IV&V, agile, and research backgrounds, will offer their insights to these questions.

References

- [1] O. Gotel, and A. Finkelstein, “An Analysis of the Requirements Traceability Problem,” *1st International Conference on Requirements Eng.*, 1994, pp. 94-101.
- [2] B. Ramesh, and M. Jarke, “Toward Reference Models for Requirements Traceability”, *IEEE Transactions on Software Engineering*, Vol. 27, No. 1, Jan. 2001, pp. 58-92.
- [3] R. Domges and K. Pohl, “Adapting Traceability Environments to Project Specific Needs”, *Communications of the ACM*, Vol. 41, No. 12, 1998, pp. 55-62.
- [4] J. Huffman Hayes, A. Dekhtyar, and J. Osborne, “Improving Requirements Tracing via Information Retrieval”, *IEEE International Requirements Engineering Conference*, Monterey, CA, Sept. 2003. pp. 138-150.
- [5] Jane Cleland-Huang, Raffaella Settini, Chuan Duan, Xuchang Zou, “Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability”, *13th IEEE Intn’l Conf. on Requirements Engineering (RE 2005)*, 29 August - 2 September 2005, Paris, France, pp. 135-144.
- [6] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia and E. Merlo. Recovering Traceability Links between Code and Documentation, *IEEE Transactions on Software Engineering*. Vol. 28, No. 10, 2002, pp. 970-983.
- [7] Center of Excellence in Traceability. <http://www.traceabilitycenter.org>
- [8] Agile Manifesto, <http://www.agilemanifesto.org>
- [9] B. Appleton, “The Trouble with Tracing: Traceability Dissected”, *CM/Crossroads, The Configuration Management Community*, http://www.cmcrossroads.com/component/option,com_magazine/func,show_article/id,94/