
RE-writings

Traceability – Problems in a Word

Dr Olly Gotel, Pace University

ogotel@pace.edu

Traceability is a topic that has been discussed and written about within software engineering circles for many years. While there has been much progress since the so-called traceability problem was studied in the early 1990s [1], traceability is still perceived to be problematic today [2]. Make a casual mention of the word ‘traceability’ when amongst practitioners or researchers and you’ll hear a few sighs. The topic is one that most of us prefer to sweep under the proverbial carpet.

I was asked to speak on traceability and its issues in software development as a panelist at RE’06 (The International Requirements Engineering Conference). I used that forum as an opportunity to air, in a somewhat light-hearted manner, some of the interrelated reasons for the continued challenges in this area. The talk itself was an excellent example of one of the main issues that compounds traceability – if no permanent record is created, what is there to subsequently trace? Traceability often comes down to people and their fading memories.

So, drawing upon my now faded memory, I have written down the intentionally provocative reasons I suggested as to why traceability problems linger. It is not meant to be a rigorous academic treatment of the topic; rather, it is more an aide-mémoire for practitioners, a mnemonic of important issues to consider when putting traceability into practice. Each issue also presents unique and interesting challenges for researchers to get their teeth into.

T **Traceability is perceived as TEDIOUS.** Discovering the requirements for a system can be difficult, but it is usually interesting work. When undertaking such a task, you interact with others as an explorer of the requirements space – you are a ‘requireonaut’¹! Writing the

¹ A little bit of history about this term: When the Requirements Engineering Specialist Group of the British Computer Society was established in 1994, the first committee brainstormed potential titles for its flagship newsletter. It arrived at the title ‘Requireonautics Quarterly’ (see issue 1 [3]). The term ‘requireonaut’ was proposed to reflect the fact that requirements engineers need to explore an unknown and seemingly unbounded requirements space, and the newsletter was created to share resources to help requirements engineers in this pursuit. The newsletter was renamed ‘Requirements Quarterly’ in 2005 (see issue 36 [4]). The subtle change was intended as a

requirements in a testable manner, then structuring them as an organized whole, is clearly an intellectual activity. It is now widely recognized how critical our descriptions are to the quality of all the development work that follows. Working with the requirements to design a system is also an engaging activity that taxes the grey matter. First-rate designers get respect. However, linking chunks of information content to establish traceability is (quite frankly) dull work and universally regarded as such. Who wants to be setting up traceability when they’d rather be exploring or designing? The inevitable upshot is that any task that is considered monotonous to undertake is put to the very end of the to-do list and is liable to result in errors when eventually undertaken. I would be so bold as to claim that the quality of the resulting traceability graph is invariably associated with the mood and boredom threshold of the person(s) doing the task. Worse, we often only find out about the quality of the traceability days, months or even years later when it is finally needed. Traceability is either in need of a serious image makeover or we have to make it invisible. Negative perceptions can cripple all good traceability intentions on a project. It is an issue to be tackled head-on.

R **RESPONSIBILITY for traceability is blurred.** Ask someone whose responsibility is it to establish the traceability on a project and they will probably use the words of Douglas Adams – it is ‘Somebody Else’s Problem’² [5]. While traceability needs its champions, it is unlikely to scale in the hands of an individual, unless you are talking about a one-person project. When traceability is assumed to be the shared responsibility of all team members, as something extra they are expected to do as part of their regular job, what do you (realistically) expect to see achieved? Firstly, do we reward those who do a superb job with setting up traceability and penalize those who jeopardize its potential?

simplification, but perhaps it was also a consequence of untraceable rationale being locked away in the fading memories of a few individuals?

² “An SEP is something we can't see, or don't see, or our brain doesn't let us see, because we think that it's somebody else's problem.... The brain just edits it out, it's like a blind spot. If you look at it directly you won't see it unless you know precisely what it is. Your only hope is to catch it by surprise out of the corner of your eye.” [5]

Secondly, are they all on the same page anyway? To distribute the responsibility and move towards scale, we need to establish a framework for all to work within, and ensure agreed roles and responsibilities are undertaken. But, if we demand exacting processes and the policing of activities, the negative perceptions mentioned earlier are simply going to explode. We could learn much from how traceability is achieved in the food industry here [6]. End-to-end traceability is achieved via a simple guiding policy that all parties subscribe to. It affords the freedom to implement traceability in independent ways so long as it complies with a standard protocol. They have distilled the essence of the traceability need in their industry, and they share both the implementation responsibility and risk of failure across the entire food chain.

A The ARTIFACTS to trace do not come pre-packaged. We deal with many different types of artifact in software development (e.g., interview transcripts, UML models, code, etc.) The information content is therefore commonly represented in a variety of media (e.g., natural language text, diagrams, sound, etc.), and at differing levels of formality and granularity. Traceability thus needs to account for information chunks in many formats and at different levels of abstraction. Unfortunately, these artifacts are not always so neatly structured and packaged within an integrated environment to enable tracing. It is all too easy to set off on a traceability venture without paying sufficient prior attention to what exactly may need to be rendered traceable and in what ways. How are trace relations to be established between radically disparate artifact types anyway? The term ‘traceability meta-model’ may populate the academic literature [7], but it is not clear how pervasive the use of such a concept is in practice. We also talk about media-based traceability in the literature [8], but many practitioners are still stuck trying to do a good enough job with plain old text. We need to step back and start by thinking about the materials we are likely to be dealing with and how these can be integrated, interrelated and handled. Elaborate schemes may be overkill, but a little bit of prior thought will go a long way.

C The CREDIBILITY of traceability can be debatable. How do we know whether we can trust the results of any traceability provided? What is the coverage of the traceability graph, how up to date is it, what is included and what is excluded? If we lack confidence in the traceability we are unlikely to use it and very unlikely to maintain it. What metrics do we

have to tell us about the quality of the traceability, for individual traceability relations, extended traceability paths and for a project as a whole? What do we do to monitor and convey these measures on a project? The whole point of this enabling mechanism we call ‘traceability’ is to help people to make more informed decisions. We can’t just establish traceability on a project and expect everything to be perfect. We need to find a way to communicate confidence levels so that people can understand the basis upon which they are taking their decisions, so think about this quality dimension when instituting traceability. You are going to need it to help you with the unavoidable issue that follows below ...

E Traceability ENTROPIES. Artifacts normally evolve as a project progresses and as learning takes place. Any traceability relations that have been established therefore have a finite shelf life and the overall traceability graph tends towards entropy without dedicated ongoing maintenance. Given it is so burdensome to put traceability in place initially, one would hope that the investment be sustained. Rather, it is more likely that the traceability is left to decay. Who is responsible for traceability maintenance anyway? Who would want to be? A more significant problem is, given that we rarely monitor the quality of the traceability, who knows if we are taking important decisions based upon data that is far from credible? Maintenance has always been the very last thought in software engineering and consequently incurs immense cost. Why should it be any different with traceability? Regrettably, this is an issue that complicates any attempt to understand the return on investment from putting traceability in place. If you are going to bother with traceability, then don’t bother if you are not going to ensure the results of your endeavors remain timely. Plan your maintenance strategy well, because the researchers have got a long way to go before they find a way to make traceability maintenance fully automatic.

A Unrealistic expectations are placed on traceability AUTOMATION. The traceability promise associated with automated tools and techniques is seductive. Many organizations purchase requirements management tools to store their traceability artifacts and relations, but these tools don’t help with any of the issues mentioned above ... yet. You still need to do all that thinking and planning. The ability to recover traceability relations automatically may be the

current Holy Grail, but such techniques still demand a high quality set of artifacts to begin with and the manual filtering of results. Rather than relying on an elusive point solution, we need to place more emphasis upon traceability strategy, figuring out how to blend heterogeneous automated and human approaches, and leveraging tools to best advantage.

B **Traceability should be a BY-PRODUCT, but it just gets in the way.** Traceability should not be the goal of software development. We are in the business of delivering systems that address customer needs, and everything else we do in software development supports this primary goal. Traceability really needs to be achieved as a by-product of other engineering activities, rendering it invisible for all extents and purposes. When we develop our software engineering practices and techniques such that support for traceability is built in as a natural by-product of other tasks, we will make progress with the traceability problem. We cannot stop what is perceived to be the ‘real work’ on a Friday afternoon to ensure the traceability is put in place on a project – it has to be integral and built in from Monday morning.

I **Ambitious INTENTIONS for traceability go unproven.** Traceability is expected and even claimed to support many development-related tasks (e.g., impact analysis, validation and verification, regression testing, trade-off analysis, change management, etc.), so the intended users and use for traceability is extremely broad. However, very little serious attention has been paid to understanding the traceability stakeholders and their tasks, and even less attention has been paid to gathering data to validate how well their goals are actually being met by whatever traceability is put in place. This is a significant issue and a somewhat embarrassing one to highlight within a community that prides itself on ‘doing’ requirements. We must do our requirements for traceability, keep on doing our requirements and follow through to see how well we are actually satisfying them.

L **Little sharing of traceability LESSONS.** This issue is the obvious result of the point made above. Where are the traceability exemplars and case studies that people can learn from? What works well and what doesn’t work well, and in what contexts? Do software engineers take traceability training prior to embarking on a project? Do they consult a little red book? We are a long way from instituting best practices in traceability

because we really don’t know what these are. Or, if we do know what these are, we certainly don’t share them. There are professional certifications for testing, quality assurance and project management, so why not a competence for traceability? Given that traceability is so essential for each of these three disciplines, this situation is surprising. While the research community busies itself with sophisticated proposals, very simple benchmark practices go unshared. This is a plea for practitioners to contribute to a body of practical knowledge for others to build upon.

I **Traceability breaks-down with the slightest INTERRUPT.** It is inevitable that there will be gaps in the traceability record due to missing artifacts and missing relations. Rather than stumble at these junctures, we need to find ways to deal with a broken traceability graph and to work with incompleteness, inconsistency and inaccuracy. The World Wide Web works despite broken links as the search algorithms have been designed to work with imperfection. We need to pay as much (if not arguably more) attention to how we can handle problems with the traceability graph as we do for building it in the first place. If you are planning on an ideal world, plan again.

T **We can’t trace the TACIT ‘stuff’.** Even if we can create an environment in which every communicative exchange is recorded, we would not be privy to tacit information, those private thoughts that go unsaid and are the basis for hidden assumptions and rationale³. Exhaustive tracing demands a way to incorporate the very edges of the information network, but attempting to do this would be a Sisyphean task. A large part of the record in software development will always remain intangible and so not available for traceability purposes. One way to address this issue is to provide links into the social structure that contributed to the tangible artifacts and forged the relations. Admittedly we may be relying on faded memories again, but at least this would provide a way to track down those memory holders. Related to the issue with interrupted traces, we need to think beyond tangible artifacts when setting up traceability and consider the necessary fallbacks.

Y **Traceability? You aren’t gonna need it (YAGNI)!** A practice of eXtreme Programming is to only implement that

³ Science fiction writers would have us believe otherwise, so we should certainly keep an open mind for the future!

functionality that is needed for the current project iteration and not to put scaffolding in place for the future. Named after the justification – ‘You Aren’t Gonna Need It’ [9]. When it comes to traceability, there can be a YAGNI feeling amongst software practitioners. Perhaps this arises when elaborate traceability systems are proposed, with huge demands on people’s time, with no conception as to when and how the results will actually be used? This is further understandable when the cost / benefit equation is largely unknown, especially for those individuals doing the work. With this issue we come full circle. We are not going to make progress with addressing yet another human perception unless we make progress with many of the issues that are outlined above.

Practitioners – many of the challenges you will face in putting traceability into practice are hidden in the very letters of ‘TRACEABILITY’ itself. Figure out how you plan to address each of these twelve issues on your projects:

- Tedious – how will you overcome this image?
- Responsibility – whose is it?
- Artifacts – what are they?
- Credibility – how will you determine this?
- Entropy – how will you deal with it?
- Automation – do you recognize its limits?
- By-product – what do you expect people to do?
- Intentions – who needs it and why?
- Lessons – are you capitalizing on these?
- Interrupts – what is your contingency plan?
- Tacit – how will you reclaim some of this stuff?
- YAGNI – how will you convince others you need it and it is worth it?

Researchers – there are lots of underlying and interrelated reasons as to why traceability problems linger. To escape traceability Catch-22 we demand systems solutions from you.

References

- [1] Gotel, O.C.Z. and Finkelstein, A.C.W. An Analysis of the Requirements Traceability Problem. In *Proceedings of the 1st IEEE International Conference on Requirements Engineering*, IEEE Computer Society Press, Colorado Springs, CO (April 1994), pp.94-101.
- [2] Cleland Huang, J., Dekhtyar, A. and Huffman Hayes, J. (Eds.) *Grand Challenges in Traceability*. Center of Excellence for Traceability Technical Report COET-GCT-06-01, University of Kentucky, September 2006.
- [3] *Requireonautics Quarterly: The Newsletter of the Requirements Engineering Specialist Group of the British Computer Society*. Issue 1 (October 1994).
- [4] *Requirements Quarterly: The Newsletter of the Requirements Engineering Specialist Group of the British Computer Society*. RQ36 (June 2005).
- [5] Adams, D. *Life, the Universe and Everything*, Pan Macmillan, 1982. See also: http://en.wikipedia.org/wiki/Somebody_Else's_Problem (accessed August 2008).
- [6] Gotel, O.C.Z. and Morris, S.J. From farm to fork or a bite of the unknown: Learning from the food industry. In *Proceedings of the International Symposium on Grand Challenges in Traceability (GCT'07), Traceability in Emerging Forms of Software Engineering*. Lexington, Kentucky, 22-23 March 2007.
- [7] Ramesh, B. and Jarke, M. Toward Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering*, 27, 1 (January 2001), pp.58-93.
- [8] Gotel, O.C.Z. and Morris, S.J. Macro-Level Traceability via Media Transformations. In *Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'08)*. Montpellier, France, 16-17 June 2008.
- [9] *Extreme Programming Practices: You Aren't Gonna Need It*. See: <http://c2.com/cgi/wiki?YouArenGonnaNeedIt> (accessed August 2008).

RE-flections

Requirements Engineering 2008 Barcelona

Ian Alexander, Scenario Plus

Ian Alexander shares his diary from RE08 in Barcelona, giving a fascinating overview of the key events and a flavour of the key issues in Requirements Engineering.

Monday

I gave a tutorial on **Rationale Modelling**, a neglected Cinderella sitting in a dark corner of Requirements practice. It was clear that, despite the varied backgrounds of the audience, everyone agreed that rationale was scarcely being captured on many projects.

Some projects (about 1 in 6, from this very small non-random survey) do model their goals; rather more model scenarios in some form; and most make some