Traceability-Why do it?

Gilbert Regan, Fergal Mc Caffery, Kevin Mc Daid and Derek Flood

Regulated Software Research Group, Department of Computing & Mathematics, Dundalk Institute of Technology & Lero,
Dundalk Co. Louth, Ireland

{gilbert.regan, fergal.mccaffery, kevin.mcdaid, derek.flood} @dkit.ie

Abstract. Traceability of software artifacts, from requirements to design and through implementation and quality assurance, has long been promoted by the research and expert practitioner communities. However, evidence indicates that, with the exception of those operating in the safety critical domain, few software companies choose to implement traceability processes, in the most part due to cost and complexity issues.
This paper presents a review of traceability literature including the implementation of traceability in real organizations. Through both analyzing case studies and research published by leading traceability researchers, this paper synthesizes the motivations of the organizations for implementing traceability. Given the importance of traceability in the regulated domain of safety critical software, the paper compares the motivations and benefits for organizations operating inside and outside of this domain.
Finally, based on an analysis of the disparate case studies, the paper re-assesses the value of traceability motivators for more widespread adoption by firms outside of the safety critical sector.

Keywords: traceability

1. Introduction

Software systems are becoming increasingly complex. Artefacts such as test cases, requirements documents, source code, design documents, bug reports etc, and the links between them are created over long periods of time by different people. Creating and maintaining these links is a difficult and expensive task. Therefore most existing software systems lack explicit traceability links between artefacts [1]. Though the

importance and role of traceability in supporting systems development have been long recognised, there are wide variations in the quality and usefulness of the practice of traceability [2].
Traceability was initially used to trace requirements from their source to implementation and test, but now plays an increasing role in defect management, change management and project management. Increasingly software development is globally distributed across multiple teams and sites which makes traceability even more relevant [3].
Good traceability information is also important to process improvement. It is fundamental for change impact analysis, requirements validation and regression testing among others. Often the quality of this information is poor, or out of date due to not being properly maintained [4].
Traceability techniques and tools are not widely used in industry [5, 6]. Complexity and cost are two of the reasons why this is the case. Companies who do adopt traceability techniques often adopt inefficient manual traceability methods and tools despite semi-automated and automated approaches becoming available [6].
This paper considers the motivations for implementing traceability. The focus is on detailing the motivators for an organisation to implement traceability and to consider if there is any difference in motivations between the general and safety critical domains.
To achieve this, a literature review including eight case studies was conducted for both generic and safety critical software domains.
This paper is structured as follows: Section 2 describes what is meant by the term traceability while section 3 details considerations when implementing traceability. Section 4 presents the research methodology used in this work. Section 5 presents the findings of this work. Our conclusion is provided in section 6 while section 7 details a future direction for this work.

2. Traceability-What is it?

There are misconceptions as to what is traceability. Section 2.1 provides a definition of traceability and an explanation of that definition.

2.1 Definition

In engineering terms a trace is comprised of a source artefact, a target artefact and the link between them [7]. Traceability therefore is the ability to establish and use these traces.
A distinction should be made between the terms Requirements traceability, Software traceability and System traceability. Requirements traceability facilitates tasks such as requirements validation and verification and focuses on tracing requirements related artifacts, using links that expose both requirements derivation and coverage [7].

Numerous definitions for traceability exist in the literature but one of the most popular and encompassing is:

> *"Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins through its development and specification to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases "[8]*.

Tracing can be performed in a forwards or backwards direction. Forwards tracing traces the requirement from its source through specification, implementation and testing. The main aim of forward tracing is to ensure that the requirement set is complete and that every requirement gets implemented and tested. It ensures that the right product is being built. Backwards tracing is tracing test cases or code back through design to the requirement specification and further onto the origin of the requirement. This can be especially useful for analyzing the impact on a requirement or its origin from a proposed change to a piece of code or test case. It helps to eliminate 'gold plating' i.e. no code or test case is written that doesn't trace to a requirement or that no requirement is written that doesn't trace to a business need, user need, standard etc
It is worth highlighting the term 'iterative' in the above definition. Stakeholders frequently introduce and change requirements therefore the continued maintenance of traces is vitally important.
Software traceability extends the definition to encompass and inter-relate any uniquely identifiable software engineering artifact to any other, extending the lifecycle coverage of the validation and verification activities accordingly. Systems traceability goes further and interrelates systems engineering artifacts to a broad range of systems-level components, such as people, processes and hardware models [7].
In general, traceability is about understanding a design right through from the origin of the requirement to its implementation, test and maintenance. Traceability allows us to understand aspects such as to whether the customers' requirements are being met, the specific requirements that an artefact relates to, and the origins and motivation of a requirement. Traceability helps ensure that 'quality' software is developed.

3. Implementation of traceability

This section presents the considerations that should be taken into account when implementing traceability. Section 3.1 looks at different aspects to be considered while section 3.2 contemplates the different trace tools available. Finally section 3.3 examines some cost considerations.

3.1 Implementation Considerations

Important considerations when implementing traceability include Pre and Post requirements traceability, non functional requirements (in addition to functional requirements), vertical and horizontal traceability, and whether to trace manually, automatically or semi-automatically.

Pre-Requirements traceability is tracing requirements from their specification to their origin. **Post-Requirements traceability** is tracing requirements from their specification through both its development and maintenance lifecycles [8]. Pre-requirements traceability is used to demonstrate that a product meets the stakeholders' stated requirements, or that it complies with a set of government regulations. Post requirements traceability supports impact analysis and requirements validation.

The tracing of **non functional requirements** i.e. system qualities such as safety, security, and performance, in addition to functional requirements, is another important consideration when implementing traceability. In practice, many organizations either focus their traceability efforts on functional requirements or else entirely fail to implement an effective traceability process. In many organizations non-functional requirements are treated in a rather ad hoc fashion and are rarely traced [9] .

Vertical traceability is tracing artefacts at different levels of abstraction, such as tracing requirements to code. Its aim is to accommodate end to end traceability. In contrast **horizontal tracing** is tracing artefacts at the same level of abstraction such as traces between requirements or traces between versions of a particular requirement at different moments in time. This highlights dependencies between requirements [10].

Manual, Automatic or semi-automatic? Traceability can be implemented manually (by a human), automatically (via automated methods and tools) or semi automatically (combination of automated methods, tools and human activities) [1].

Traceability is implemented manually in many organizations [6] due to the cost and complexity of automation, therefore potential of software traceability is often not exploited, particularly in smaller companies.

Automatic traceability is much faster than manual tracing but in reality automatic traceability falters in either producing inexact traces or misses required traces. Traces can be produced after-the-fact or in-lifecycle. After the fact traces are of a "good enough" nature and sometimes not perfectly recovered. For in-lifecycle tracing, manual tracing is time consuming and arduous while fully automated tracing sometimes produces inaccurate traces and human analysts are reluctant to take responsibility for them. Semi-automated tracing therefore would seem to offer the best of both worlds, where human analysts make final traceability decisions.

"*When performing semi-automated tracing, human analysts, at a minimum, need to examine the results produced by the automated methods. Additionally, analysts may interact with the tracing software, provide tracing feedback to the software, and ask the tracing software to retrace*"[4].

3.2 Tool Options for Traceability

Tool options for traceability fall into three basic categories:
1. Dedicated Requirements Management Tools – which concentrate on supporting the fundamental activities of requirements management and are frequently referred to as traceability tools due to their focused support in this area. The advantage of using a dedicated requirements management tool is that it focuses exclusively on the fundamental requirements management activities and on enabling traceability [7].
2. Lifecycle Tools –support a wide span of the software and systems development lifecycle and manage its broader artifact types. The traceability provided can be more generic in nature than with the dedicated tools, though more encompassing of lifecycle phases, and a single lifecycle tool may provide for a total tooling solution. End-to-end traceability is possible, in theory. There can also be benefits from having fewer tools to learn to use and to handle [7].
3. General-purpose Tools and Proprietary Development – Text editors, graphic editors, spreadsheet tools, databases and wikis are all general-purpose tools that can all be configured to allow previously manual and paper-based requirements management activities to be carried out with some form of tool support. The advantage here is that general purpose tools are widely available and many people already know how to use them. General purpose tools are most suited for small and short lived projects. There is also the option to develop a proprietary tool completely from scratch. However tool building is not the primary focus of most organizations and is sometimes best left to those with expertise in the area [7].

3.3 Cost Considerations

Cost, as always, is an important factor. An optimal balance of cost and trace quality can be achieved by ranking requirements and refining trace quality as necessary. A traceability strategy can combine tactics such as; varying the granularity of trace links, varying the coverage of trace links, adopting tools where possible to optimize recall or precision, varying the frequency of trace link maintenance [11].
A flexible approach to choosing techniques is prudent. Prioritizing requirements means a selection of techniques can be used thus ensuring the best balance between cost and quality. However "*a 'greedy' approach would result in each requirement using the technique that best supported its own needs. Such an approach could result in the excessive use of links at the project level, and create a non sustainable situation. Individual traceability decisions must therefore be made within the context of project level trace objectives*" [12].

4. Research Methodology

The purpose of this research is to answer the following research questions:

RQ 1: What are the motivators for an organisation to implement traceability?

In addition to examining the motivators, we also decided to investigate any difference in motivations between the general and safety critical domains?

The literature on traceability was surveyed to gather viewpoints from both research experts and industry practitioners regarding the potential motivations of implementing traceability to an organisation.

The portals that facilitated the research were IEEE Xplore digital library, ACM digital library and Google Scholar. Key search words used were *traceability*, *software+traceability*, *requirements+traceability*, *traceability + case + study*, *traceability+survey*, *traceability+motivators*. These searches returned more than 150 publications. Each abstract was scanned for relevance to our topic and 45 publications were identified. On further examination of their content, 33 were selected to inform our research, based on their relevance to our search topic. 14 of these were case studies from which we selected 8 of the more recent ones.

The book Software and Systems Traceability 2011 (Gotel, Cleland-Huang and Zisman) informed a great deal of our research. Many of the chapters from this book are referred to throughout this paper.

Details of the aforementioned case studies are:

Klimpke: 2009 [13] interviewed stakeholders in five enterprises (with different backgrounds regarding size, type of software, sector and number of locations)in relation to their experiences of using traceability. The organisations ranged from very large (with 10,000 people working in development and projects of 40,000 requirements) to small (200 people and projects of 50 to 300 requirements).

Panis: 2010 [14] carried out a traceability survey on-line at Teradyne (US manufacturer of ATE). 23 engineers (4 Systems Engineers, 7 Subsystem Engineers, and 12 Design Verification Engineers) responded directly to the survey regarding their experiences with traceability.

Arkley and Riddle: 2006 [15] observed a project, conducted at BAE SYSTEMS Electronics and Integrated Solutions (E&IS) (Plymouth, UK) that developed a requirements traceability system which is integral to their development process. The E&IS operating group of companies designs, develops and manufactures a wide range of electronic systems and subsystems for both military and commercial applications.

Neumuller and Grunbacher: 2006 [6] introduced traceability into a very small Austrian software company (GeDV2), whose main product is a business information system for small and medium-sized enterprises. The product is used by 29 customers; the largest installation supports about 150 concurrent users. To implement traceability GeDV developed a number of customised tools in-house and they also established some fairly simple coding and id conventions.

Mc Caffery: 2011 [16] assessed two SME companies (one in Ireland and one in the UK) who operate in the medical device domain. Both companies develop electronic

based medical devices that are marketed in the US and Europe. To sell their products they require compliance with both the FDA and the MDD.
In both organizations the importance traceability plays in medical device software development was understood and a member of the management team was responsible for its implementation. The dual role of tracing requirements and managing risk and hazards were appreciated, but were recognized as complex and difficult to achieve.
Heindl and Biffl: 2005 [17] reports a case study on value-based requirements tracing (VBRT) that systematically supports project managers in tailoring requirements tracing precision and effort, based on the following parameters: stakeholder value, requirements risk/volatility, and tracing costs. The research question to be answered is: To what extent can VBRT reduce requirements tracing efforts (economy of requirements tracing)? The case study project "public transport on demand" is about an improved and more efficient public transportation system in rural areas supported with modern information technologies. The challenge is to stop further deterioration of public transportation access in rural areas with a new traffic model.
Born, Favaro and Kath: 2010 [18] report on experience gained with the application of ISO 26262 (international standard for functional safety of road vehicles) in a pilot project at a German car manufacturer as well as experience from various consultancy projects.
Mader, Gotel and Philippow: 2009 [19] reports on an exploratory study of the traceability practice within ten companies based predominantly in Germany. Only two of the ten practitioners selected were known to have a prior interest in traceability topics. The size of the participating companies included six medium (50–500 employees) and three large (>500 employees) companies. The only small company (<50 employees) actually reported about a consulting project they were undertaking for a large company and the traceability practices therein. "Our cases included a mix of software development offering, including companies who predominantly create end products to sell to a user, who do project development work for other companies supplying a market, or who provide expert advice on processes, techniques and methods. Most worked in the transportation domain (avionics and automotive). The subjects we interviewed held the following positions: three system analysts, two consultants, one requirements engineer and four team or project leaders" [19].

5. Findings

While recognising that there are many obstacles to discourage an organisation from implementing traceability, this section considers the factors/reasons why an organisation would benefit from implementing traceability. This section synthesises industry viewpoints on motivation (taken from the case studies) with those published by established researchers. This section provides an answer to RQ 1: What are the motivators for an organisation to implement traceability?
Regulation: Traceability implementation is mandated in many software development standards as seen in [2, 6, 13, 14, 16, 19-21], and many industries, in particular the

safety critical industries e.g. in the US the Federal Drugs Authority states that code must be linked to requirements and test cases. In Europe, a medical device cannot be marketed unless it is developed using processes that comply with the European Council's Medical Device Directive (1993/42/EEC) [22] and amendment MDD (2007/47/EC) [23]. To be marketed in the US, a medical device must be developed using processes that comply with Federal Drugs Authority guidelines. "In both locations medical device companies must be able to produce sufficient evidence to support their compliance"[16].

Safety Case: Safety critical systems must satisfy a number of non-functional requirements e.g. safety, availability and reliability [16, 18]. Regulation normally requires critical systems are certified before entering service. This involves submission of a safety case - a reasoned argument and supporting evidence that such requirements have been met and that the system is acceptably safe" [24]. A safety critical organisation who fails to make a safety case can be fined or have its product recalled. These organisations must employ bi-directional traceability such that the safety case shows full life-cycle traceability between hazards, requirements, design code and test cases [25]. A good safety case encompasses an effective risk mitigation process which is highly dependent on requirements traceability. It should also be noted that a good safety case is useful in defence of any litigation.

Competitive Advantage: Traceability reduces production costs [15, 17] through reuse, error avoidance etc. Traceability also allows for impact analyses (case studies [6, 13-16]and general literature [9, 26, 27]) which in turn allows for cost estimation of any change to a system. A change to any artefact will impact on other artefacts. This is particularly useful during the maintenance phase when the cost proposed changes needed to be communicated to the relevant stakeholders. The cost of achieving each quality goal and the recognition of risk is more completely understood by stakeholders through the ability to trace functional and non-functional requirements [28]. Traceability improves the ability to develop realistic cost proposals and gain competitive advantage in building similar systems due to "savings from using [a] lessons learned database of critical issues and rationale"[2].

Productivity and Quality gains: Traceability helps facilitate productivity and quality gains [13, 15-17, 19]. Reuse of 'proven' artefacts from design, code or test stage ensure productivity gains as these artefacts don't have to be reproduced, saving both time and effort. As they have already been proven, quality and reliability should be assured [29]. In addition to this there are occasions when a requirement should not be changed. This may be because of regulations; therefore linking requirements to sources can help avoid conflict and rework.

Requirements validation: Process conformance will be facilitated through traceability as requirements validation [25] will be ensured and the product will satisfy customer requirements [14, 15, 17, 19]. While unidirectional forward tracing facilitates requirements validation, the reverse of this(tracing from code to design and onto requirements) helps mitigate the risk of 'gold-plating' i.e. excess artefacts or functionality [25].

Identification of stakeholders: Traceability helps project leaders identify the relevant stakeholders [6, 15] to involve when drawing out requirements and makes the requirements negotiation stage easier as backward tracing links requirements to their origin.

Rationale for decisions: Many critical decisions are taken during the SDLC and the rationale for these decisions [13, 14] can be lost unless they are documented and traced to the corresponding artefacts. This facilitates new members learning (Mc Caffery [16] highlighted training) and is also useful because, after a period of time, developers may not remember the rationale behind decisions. Rationale is helpful for handling major system extensions, refactoring or preparing safety cases [11].

Change management: Sometimes it is cheaper to build software quickly and change it if it fails to meet requirements [1] (maybe because of volatile requirements) as in the Agile process. Constant change of requirements [13-16, 19] means a greater need for traceability support. It also means greater customer interaction. Traceability can thus allow developers to better manage customer relationships [6, 15]. Traceability tracks requirements (or user stories) to versions [18, 19] i.e. to track exactly which requirement has been implemented in a specific version. Traceability also tracks dependencies between requirement i.e. requirements to requirements [25].

Project and Risk management: The role traceability plays has expanded and it has become an important tool in the software development activities of project management, risk management, and defect management [14-17]. This is particularly relevant as software development is increasingly globally distributed across multiple teams and sites [3]. It is therefore essential to have an effective traceability process in place as it provides an essential support for developing high quality software systems [3].Traceability provides confidence [14].

Variability in Product Line Engineering: In PLE traceability helps to understand dependencies among diverse reusable artefacts as well as between the product line and the derived products which often include additional developments and customizations. Traceability in PLE thus helps understanding variability and ensuring the consistency of products. Engineers need traceability support in IDEs when modifying product line artefacts[29].

Other motivators for traceability include the ability to measure test coverage [11], test success [13], project progress [13, 30], reduction in requirements creep [11, 30] easier program understanding [6]. Finally, traceability facilitates code maintenance [6, 17, 31].

Regulation and safety case are the two main differences in motivations between the safety critical domain and general domain. These two motivators are particularly relevant to the safety critical domain. The remaining motivators are relevant to the safety critical and general domains. Organisations operating in the safety critical domain are mandated by regulations to implement traceability. They must provide acceptable evidence that their system is safe and that risks have been mitigated and be able to prove a safety case as failure in a safety critical domain can cause great harm or even loss of life.

6. Conclusions

While we recognise that there are many barriers to implementing traceability (such as cost, tooling issues, trace decay, difficulties in tracing NFRs and lack of implementation guidance), the focus of this paper is on the motivations for an organisation to implement it. The research question which guided this work was: 'What are the motivations for a company to implement traceability?' In addition to that question we also decided to investigate any difference in motivations between the general and safety critical domains.

According to the literature, traceability is beneficial. However its implementation is inconsistent at best, with most companies not implementing it or implementing it in a haphazard manner. As this paper shows, there are many advantages to a company implementing and using traceability with productivity, maintenance and quality gains being chief among them. These benefits, along with reductions in production costs, help give an organisation a competitive advantage. Customer satisfaction is enhanced through requirements validation. Risk is better managed as it is crucial to maintain traceability between the software safety requirements, the decisions taken during design and their actual implementation in the code. Impact analysis allows for better change management which is especially useful when requirements are volatile.

All of the above motivators are important to both the general and safety critical domains. However for the safety critical domain, 'regulation' and 'safety case' are two extremely important motivators.

7. Future Work

While this paper presents the motivations for an organisation to implement traceability, it does not describe the barriers that an organisation might face in implementing traceability. Important future work would be to describe these barriers and how to overcome them.

It is noted that the amount of information in literature regarding traceability implementation is limited. We also plan to analyse the current state of practice for software traceability within Irish companies with a view to providing a framework for the implementation and use of traceability.

Acknowledgements

This research is supported by the Science Foundation Ireland (SFI) Stokes Lectureship Programme, grant number 07/SK/I1299, the SFI Principal Investigator Programme, grant number 08/IN.1/I2030 (the funding of this project was awarded by Science Foundation Ireland under a co-funding initiative by the Irish Government and European Regional Development Fund), and supported in part by Lero - the Irish Software Engineering Research Centre (http://www.lero.ie) grant 03/CE2/I303_1.

8. References

[1] Gotel, O., *et al.*, *Software and Systems Traceability*. London Dordrecht Heidelberg New York: Springer, 2011.

[2] Ramesh, B., "Factors influencing requirements traceability practice," *Commun. ACM*, vol. 41, pp. 37-44, 1998.

[3] McCaffery, F., *et al.*, "Medical Device Software Traceability," in *Software and Systems Traceability*, O. G. e. al, Ed., ed: Springer, 2011.

[4] A.Dekhtar and Hayes, J. H., "Studying the role of Humans in the Traceability Loop," in *Software and Systems Traceability*, ed: Springer, 2011.

[5] Gotel, O., *et al.*, "The Grand Challenge of Traceability," in *Software and Systems Traceability*, ed: Springer, 2011.

[6] Neumuller, C. and Grunbacher, P., "Automating Software Traceability in Very Small Companies: A Case Study and Lessons Learne," in *Automated Software Engineering, 2006. ASE '06. 21st IEEE/ACM International Conference on*, 2006, pp. 145-156.

[7] Gotel, O. and Mader, P., "Acquiring Tool Support for Traceability," in *Software and Systems Traceability*, ed: Springer, 2011.

[8] Gotel, O. C. Z. and Finkelstein, C. W., "An analysis of the requirements traceability problem," in *Requirements Engineering, 1994., Proceedings of the First International Conference on*, 1994, pp. 94-101.

[9] Cleland-Huang, J., "Toward improved traceability of non-functional requirements," presented at the Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering, Long Beach, California, 2005.

[10] Gotel, O., " Traceability Fundamentals," in *Software and Systems Traceability*, O. G. e. al, Ed., ed: Springer, 2011.

[11] Ingram, C. and Riddle, S., "Cost Benefits of Traceability," in *Software and Systems Traceability*, ed: Springer, 2011.

[12] Cleland-Huang, J., *et al.*, "A heterogeneous solution for improving the return on investment of requirements traceability," in *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, 2004, pp. 230-239.

[13] Klimpke, L. and Hildenbrand, T., "Towards End-to-End Traceability: Insights and Implications from Five Case Studies," presented at the Proceedings of the 2009 Fourth International Conference on Software Engineering Advances, 2009.

[14] Panis, M. C., "Successful Deployment of Requirements Traceability in a Commercial Engineering Organization...Really," in *Requirements Engineering Conference (RE), 2010 18th IEEE International*, 2010, pp. 303-307.

[15] Arkley, P. and Riddle, S., "Tailoring Traceability Information to Business Needs," in *Requirements Engineering, 14th IEEE International Conference*, 2006, pp. 239-244.

[16] Caffery, F. M. and Casey, V., " Med-Trace " presented at the SPICE 2011, 2011.

[17] Heindl, M. and Biffl, S., "A case study on value-based requirements tracing," presented at the Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering, Lisbon, Portugal, 2005.

[18] Born, M., *et al.*, "Application of ISO DIS 26262 in Practice," presented at the CARS, Valencia,Spain, 2010.

[19] Mader, P., *et al.*, "Motivation Matters in the Traceability Trenches," presented at the Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, RE, 2009.

[20] McCaffery, F., *et al.*, " How Can Software SMEs Become Medical Device Software SMEs " presented at the EuroSPI 2011, 2011.

[21] Asuncion, H. U., *et al.*, "An end-to-end industrial software traceability tool," presented at the Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, Dubrovnik, Croatia, 2007.

[22] Council, E., "COUNCIL DIRECTIVE 93/42/EEC of 14 June 1993 concerning medical devices," ed. Official Journal of The European Communities, Luxembourg (1993), 1993.

[23] Council, E., "DIRECTIVE 2007/47/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 5 September 2007," in *Official Journal of the European Union*, ed. Luxembourg, 2007.

[24] Mason, P., "On Traceability for Safety Critical Systems Engineering," presented at the Proceedings of the 12th Asia-Pacific Software Engineering Conference, 2005.

[25] Cleland-Huang, J., "Traceability in Agile Projects," in *Software and Systems Traceability*, ed: Springer, 2011.

[26] Hayes, J. H., *et al.*, "REquirements Tracing On target (RETRO): Improving Software Maintenance through Traceability Recovery," 2007.

[27] Antoniol, G., *et al.*, "Recovering Traceability Links between Code and Documentation," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,*, vol. VOL. 28, 2002.

[28] Kazman, R., *et al.*, "ATAM: Method for Architecture Evaluation," Carnegie Mellon Software Engineering Institute, Pittsburgh2000.

[29] Heider, W., *et al.*, "Evolution-Driven Trace Acquisition in Eclipse-Based Product LineWorkspaces," in *Software and Systems Traceability*, ed: Springer, 2011.

[30] Arkley, P. and Riddle, S., "Overcoming the Traceability Benefit Problem," presented at the Proceedings of the 13th IEEE International Conference on Requirements Engineering, 2005.

[31] Oliveto, R., *et al.*, "Software Artefact Traceability: the Never-Ending Challenge," in *Software Maintenance, 2007. ICSM 2007. IEEE International Conference on*, 2007, pp. 485-488.