Adapting Traceability in Project Lifecycle

Palak Sharma

Master's Student at Software Engineering Department Rochester
Institute of Technology. Rochester, NY. 14623. USA
ps2671@g.rit.edu

***Abstract*— Software traceability refers to identifying the interconnection between different software artifacts such as requirements, source code, UML diagrams and test Cases. The focus of this survey paper is to look into traceability techniques related to software process modeling. In particular, four research papers are taken into consideration, with emphasis on customized traceability frameworks for projects and the future work to improve their efficiency. Domges and Pohl (1998) looked into the importance of customizing software traceability for project specific needs, for improving cost benefit ratio. Letelier (2002) developed a traceability framework which can be adapted as per customized needs for UML based projects. Kelleher (2005) propose a traceability metamodel (TRAM) and process (TRAP) to create process lifecycle in form of web application from the stakeholders' perspective. Cleland-Huang (2014) dwells deep into the earlier, current and future trends in software traceability field and puts forth several intriguing research questions to be addressed.**

I.INTRODUCTION

Spanoudakis and Zisman (2005) defined Software Traceability as *"the ability to relate artefacts created during the development of a software system to describe the system from different perspectives and levels of abstraction with each other, the stakeholders that have contributed to the creation of the artefacts, and the rationale that explains the form of the artefacts"*. Some of the major applications of software traceability are in Impact Analysis (updating non-code project artefacts) and Requirement tracing (handling new change requests to the system).However more recent trends in this field focus on *"model driven development, agile project environments and product line systems"* (Cleland-Huang et al., 2014).

Domges et al (1998) looked into the effectiveness of software traceability in terms of project metrics - cost and time - and other external factors which influence the trace capture and usage. By taking into account stakeholder's perspective, they propose trace data strategies for project specific needs. In order to overcome the project specific issues with existing traceability frameworks, the authors also proposed a project specific traceability framework "PRIME-RT" and implemented it on two prototypes : "*PRO-ART and TECH-MOD*".

An addition to project specific traceability framework, Letelier (2002) proposed a reference metamodel for requirement traceability that makes use of UML models and demonstrate it on a small- sized "Rational Unified Process" (RUP) project.

In order to make traceability framework reusable, Kelleher (2005) introduced a novel traceability metamodel and process which can be applied to any product lifecycle.

The rest of the paper is organized as follows : (II) Background - Software Traceability Techniques, (III) PRIME-RT: Project-Specific Traceability Framework, (IV) Reusable Traceability Framework, (V) Traceability Framework applied to RUP, (VI) Future Trends and challenges, (VII) Conclusion, (VIII) References.

II. BACKGROUND: SOFTWARE TRACEABILITY

Software traceability techniques can be broadly categorized under two heads: semantics and syntactic. The syntactic techniques make use of text matching between the different pair of artefact to establish a link between the two. However semantic techniques involves use of machine learning algorithms to understand the domain specific connection between the artefacts instead of finding textual similarity between them. Information retrieval techniques - used to determine similarity between artefacts - are based on factors such as text frequency, position and distance between the words.

Some of the commonly used Information retrieval techniques are Vector Space Model (VSM), Latent Semantic Indexing (LSI), and Latent Dirichlet allocation (LDA).VSM is an algebraic model to represent document and query in terms of vectors. The similarity between the document and the query is determined by calculating the cosine angle between the document and query vector (Salton et al., 1975). LSI works on the principle of conceptual correlations between the words used in same context. A weighted term-document matrix is created and analyzed to identify the related concept in the text (Deerwester et al., 1990).LDA is a statistical model used for determining similarity in natural language processing. It assumes that each document is defined in terms of particular set of topics depending on the context of the words usage. (Blei et al., 2003).

III. PRIME-RT: A PROJECT-SPECIFIC TRACEABILITY FRAMEWORK

As each project has its own specific requirements, the traceability environment should not only relate artefacts but also allow the users to add project customized data types and strategies and also provide *"Integrated method guidance"* for stakeholders. Other important factor impacting the efficiency of traceability for project use is the *"organizational learning"* factor. To accomplish this, Domges et al (1998) proposed" *PRIME-RT: A Project-Specific Traceability Framework"* (Figure 1). It has three main elements:

(i) *Method Definition Environment:* The project manager specifies the project specific traceability requirements by referring to the modeling concepts from the trace repository and thus updating the details in the repository. Such an environment, promotes error reduction along with managing the project- specific definitions, and hence consistently supporting the project manager in his activities.

(ii) *Process Execution Environment:* The application Engineer has the guidance of past recorded experiences and the project-specific trace definitions specified by other stakeholders. Such a framework support the compliance of the "new traces" defined by the application engineer with the "stakeholder specified traces". Hence, the validity of the information in the trace repository is preserved and enriched.

(iii) *Method Improvement Environment:* Importance of organizational learning is a crucial element of this *"PRIME-RT"* framework. To facilitate the "method improvement environment", the framework allows the method engineer to update the trace repository with modified definitions based on his experience.

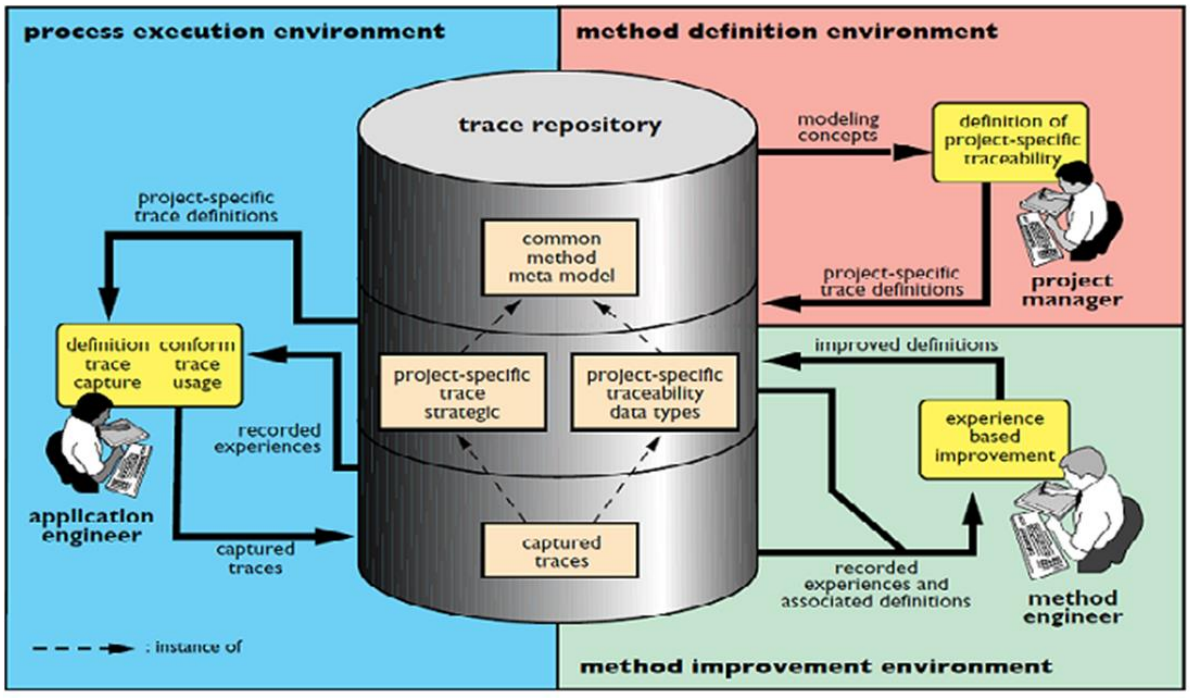

Fig. 1. Project-Specific Traceability Framework

The goal is to make the traceability process driven by the project lifecycle, history and the stakeholder objectives and hence strive for continuous improvement. To achieve this, this framework enables *"data-driven, user-driven, and process-driven strategies"* (Domges et al, 1998). Based on this framework, two prototypes were developed *"PRO-ART supporting project specific traceability in requirements engineering processes, and TECHMOD supporting project-specific traceability during the modeling of complex chemical processes"*. When analyzed on small sized case studies, the feedback verified the improvement in product quality due to use of this proposed framework.

IV. Reusable Traceability Framework

Kelleher (2005) focused on reusability aspect for traceability and developed a metamodel (TRAM) which can be used to describe any process irrespective of its domain. He used the Software Process Engineering Metamodel (SPEM) for designing the model. Figure 2 demonstrates the proposed architecture. The elements of TRAP are used to describe the process lifecycle.The TRAM libraries guide and support the process lifecycle activities. The Meta Object Facility (MOF) is an abstract language and framework built upon TRAM and TRAP.

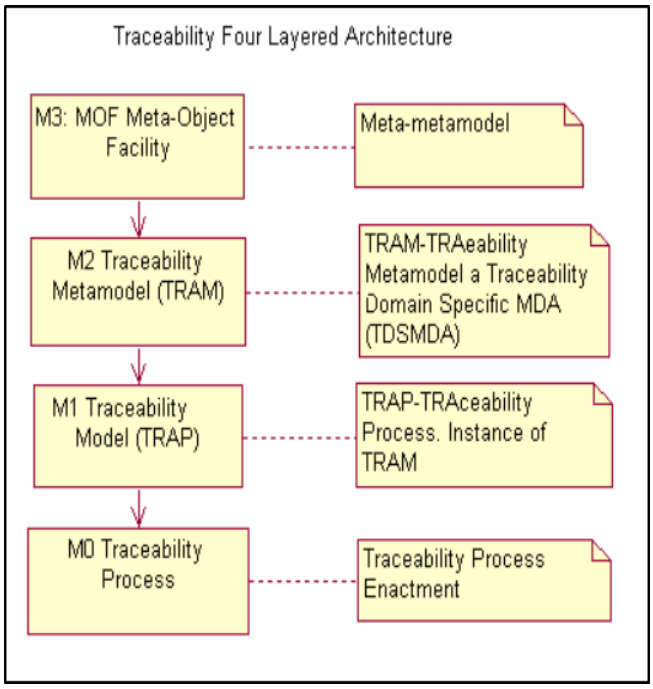

Fig. 2. The Traceability Process Architecture (Kelleher, 2005)

Traceability Process (TRAP): It is a process management tool which comprises of all the essential traceability techniques required for any product lifecycle. The process is described in terms of sequential high level workflows. These workflows are further fragmented into activities, roles and artifacts. Figure 3 shows example of a simple traceability process (TRAP) for a requirement engineer. The diagram illustrates high level artifacts such as test artifacts and design artifacts required by the engineer. The engineer tasks is to find traces between the test and design artifacts with respect to the requirement database and the available traceability matrix, guides in the trace matching task.

Traceability Metamodel (TRAM): TRAM contains all core and extension packages .The core packages contain best practices in form of guidance meta-class and the extension package contains all dependencies related elements.

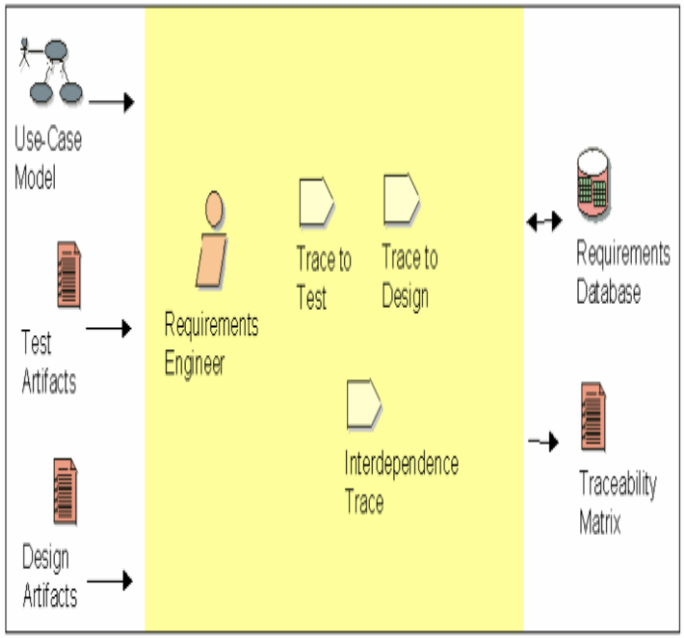

Fig. 3. TRAP Process for a Requirement Engineer (Kelleher, 2005)

V. Traceability Framework applied to RUP

Letelier (2002) presented a customized traceability framework for project specific needs by interconnecting textual specifications with UML models. The author developed requirement traceability model as depicted in figure 4. This model covers the pre-traceability (origins to requirement) and post traceability (Requirement to code) aspects.

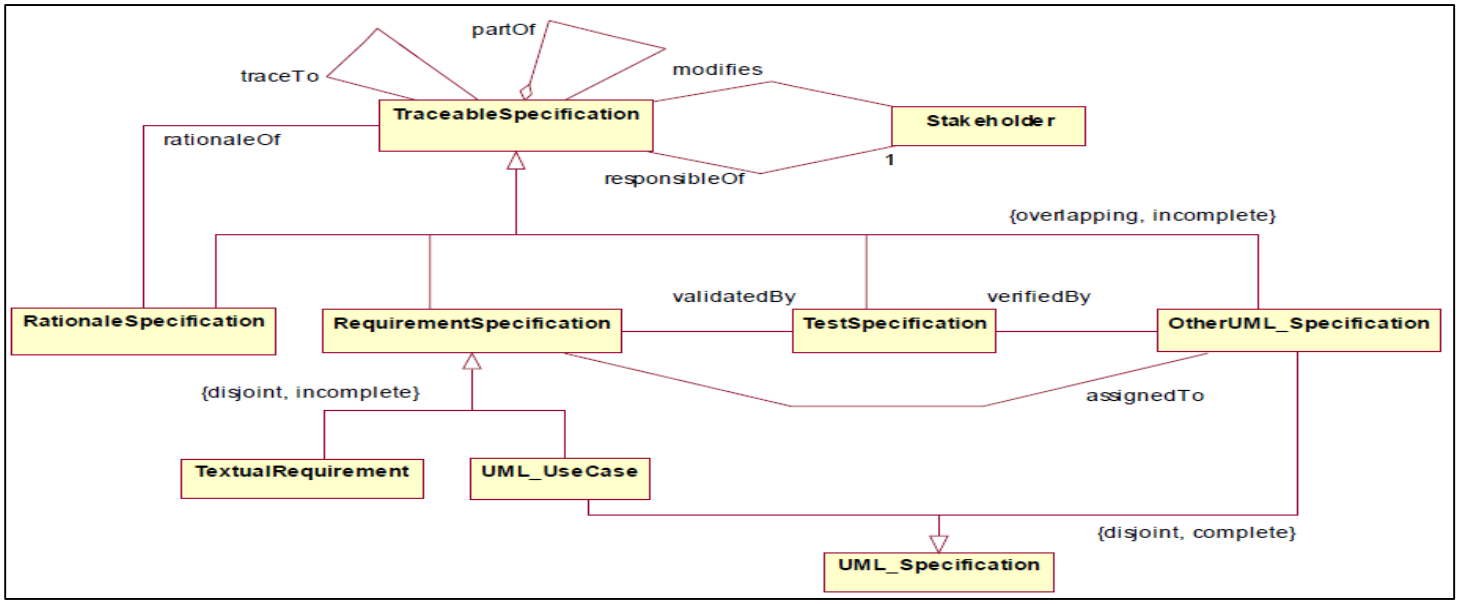

Fig. 4. Requirement Traceability Metamodel (Letelier, 2002)

The classes represent entities and the associates indicate the traceability links *(modifies, responsibleOf, rationaleOf, validatedBy, verifiedBy and assignedTo)* between the entities. On a high level two entities can be seen, the "*TraceableSpecification and the Stakeholders*". *Stakeholders* are responsible for managing a specification, while the *TraceableSpecification* has different granularity level depending on its type. It can be of type *"RationaleSpecification, RequirementSpecification, TestSpecification, and OtherUML_Specification"*. The child entities of *"TraceableSpecification and RequirementSpecification"* are defined as *incomplete* so that information from other sources can be collaborated with them and the final complete model is created.

Letelier applied the proposed model to a small RUP project as indicated in figure 5. The graphs represents the traceability between artifacts in case of simple "customer registration". FEATURE means Software Feature, UC means Use Case, UCS means Use Case Specification, PRECOND means Use Case Precondition. The traceability links are derived by the aggregation relationship between the Software feature *Sales management and Book sales*. The link from *UC to PRECOND* is derived by name matching.

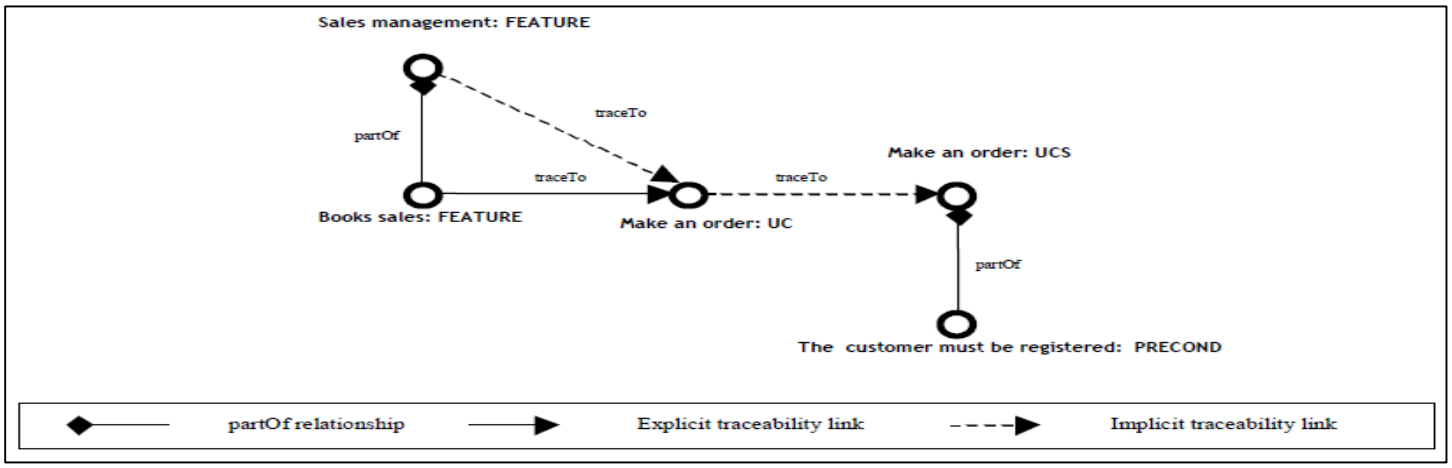

Fig. 5. Traceability Model applied to RUP (Letelier, 2002)

VI. Future Trends and Challenges

Cleland-Huang et. al (2014) analyzed the earlier and current trends in the field of software traceability and points out some intriguing future research questions concerning the *cost-effectiveness, trusted, scalable, portable, ubiquitous*, and *visualization* aspects of traceability techniques.

There is strong need to create human centric tools which promote trace detection as well as ease to visualization. Another area of focus is to develop automated traceability algorithms to validate the correctness of the recorded traces. Instead of creating whole together new tools, more beneficial approach is to develop flexible tools which can be integrated with development environment.

VII. Conclusion

This paper summarizes few research studies related to adapting traceability framework on product lifecycle as a whole. In particular, traceability for project specific requirement is the prime focus of this study. Earlier studies by Domges et. al (1998) emphasizes on the value of project specific traceability and how to incorporate it by continuous improvement process for the benefit of the stakeholders. Kelleher (2005) proposed a UML based framework to create traceability visualizations for any project. On similar grounds, Letelier (2002) presents a traceability model which can be applied to any project which make use of UML Modelling.

VIII. References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 993-1022.

Cleland-Huang, J., Gotel, O. C., Huffman Hayes, J., Mäder, P., & Zisman, A. (2014, May). Software traceability: trends and future directions. In Proceedings of the on Future of Software Engineering (pp. 55-69). ACM.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6), 391.

Dömges, R., & Pohl, K. (1998). Adapting traceability environments to project-specific needs. Communications of the ACM, 41(12), 54-62.

Kelleher, J. (2005, November). A reusable traceability framework using patterns. In Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering (pp. 50-55). ACM.

Letelier, P. (2002, September). A framework for requirements traceability in UML-based projects. In Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering (pp. 30-41).

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613-620.