The Need for a Unifying Traceability Scheme

Angelina E. Limón and Juan Garbajosa

Technical University of Madrid (UPM, *Universidad Politécnica de Madrid*),
Mail address: E.U. Informática. Ctra. de Valencia Km. 7,
28031 Madrid, Spain
aespinoza@zipi.fi.upm.es
jgs@eui.upm.es
http://syst.eui.upm.es

Abstract. The benefits of traceability are widely accepted nowadays, however, several issues still make it difficult a wide-scale adoption of traceability in the software engineering practice. There is a lack of a commonly accepted traceability definition further than the term definition, a standard way of specifying traceability among items, and a traceability type classification; besides, conflicts among a number of approaches exist. As a result traceability-schemes implementation in tools lacks of generality and exchangeability. Round trip engineering therefore cannot be well enough supported. The motivation behind is aligned with that of PIM within the MDA initiative. This paper analyzes several current traceability schemes, in order to obtain relevant features and identify overlaps and inconsistencies among the approaches. Then, and based on the analysis, it provides an initial approach for a Traceability Specification Scheme. This scheme is expected to facilitate traceability specification for a given project, to improve the traceability management, and help to automate some traces management processes.

1 Introduction and Motivation

Traceability is widely recognized as a concern in software and systems engineering, this is reflected by an extensive literature, multiple tools, and a growing research interest in the area [15]. The benefits of a well managed traceability activity are widely accepted nowadays, however, several issues still make difficult the adoption of a wide-scale traceability activity in the software\system engineering practice. There is a lack of a commonly accepted traceability definition, further than the scope of the term definition as in [2]; also a standard way of specifying traceability among items does not exist. It is interesting to notice that standards such as [9] and [10], widely used in industry, do not provide a definition for traceability. A broadly accepted traceability type classification also is missing. Then, traceability in many organizations is haphazard, the standards provide little guidance, and the models and mechanisms vary to a large degree and are often poorly understood [15].

Considering this landscape, the final motivation for this research work is to provide a traceability scheme general enough to support those features *useful* for

roundtrip engineering but simple enough so that it can be easily implemented in tools and used to exchange traceability information among different model representations such as in the MDA case. Furthermore the philosophy that underlying this work and that of PMI are somehow similar. Both of them search for rather independent representations that work as an umbrella either for platform dependent (MDA) or, in our case, specific traceability models.

An essential input for the work is the analysis of the current available literature since the traceability scheme should gather those common features of the traceability approaches analyzed, together with some proposals, in order to improve the process control of the traceability management.

Under the MDA approach advantage of the models can be taken through roundtrip engineering between an abstract model of the system and describing the system architecture and or design, and the code [3]. Roundtrip is also as necessary in verification and validation processes and non-conformance management [9]. To provide automated support to these processes is one of the high level objectives of the research group where this work has been performed including general purpose software engineering environments and validation environments such as TOPEN [1].

Section 2 of this paper, *Analysis of Traceability Approaches* presents a summary of the analysis of several traceability proposals. Section 3, *An Initial Proposal for a Traceability Scheme Specification* is introduced, and finally in section 4, a number of conclusions are provided.

2 Analysis of Traceability Approaches

One of the aims of the analysis has been to identify a number of features that are the baseline for the available literature. The following features will be the starting point for the analysis and assessment and will be studied more in depth in the following subsections.

1. The link may be *process-related* or *product-related* (concepts defined by Jarke and Ramesh [15]).
2. The *Pre-RS*[1] and *Post-RS* traceability relations categories, as defined by Gotel and Finkelstein [8])
3. The traceability link *purpose* introduced by Jarke and Ramesh [15]). The link purpose must provide information on the reason or justification of the link existence as project artifact, in order to justify the project resources that will be spent on the link (resources such as, for example to monitor, to save, and to analyze it in order to make project decisions, etc).
4. The items or objects which the traceability link will relate. It is desirable the objects can be specifically defined, in order to make the most automatically possible the linkage tasks.

[1] Requirement Specification

2.1 Process and Product-related Link Type

- Jarke and Ramesh [15] present the *Satisfies, Dependency, Evolution and Rationale* classification. The *Satisfies* and *Dependency* links are qualified by authors, as *product-related*.
- Gotel and Finkelstein in [8] introduce the *Post-RS* links, defined to trace the requirements from and back to RS, through artifacts in which they are distributed. Therefore, it is clear that those links include the *Satisfies* and *Dependency* links which are *product-related*.
- Jarke in [11] also presents a classification which considers the *Post-RS* relations defined as relations among requirements to design and implementation. *Post-RS* links include the *Satisfies* and *Dependency* links which are *product-related*.
- Spanoudakis et al. [16] define the *Overlap* and *Requires_Feature_In* which are *Dependency* links and *Requires_Execution_Of* and *Can_Partially_Realize* which are *Satisfies* links, as authors themselves assure. Therefore, those four ones are *product-related*.
- Van Vliet et al. [12] define a three type classification, more oriented to product family traceability:
  - Traceability between *PF*[2] *feature* and *Product feature* is an *Association* relation of the *Supports* type and is divided into subtypes: *ComposedOf* (that is a *Derived* link, because the composed objects were derived from its father) and *Requires-Excludes* (that is a *Dependency* link).
  - Traceability between *Product FM*[3] and *Product CM*[4] is an *Association* relation of *Realizes* type, thus is clear that is a *Satisfies* link.
  - Traceability between *Product CM* and *Implementation* is an *Association* relation of *Implements* type. Then is inferred this traceability relation is a *Satisfies* link.

  Therefore, the all three last links include the *Satisfies* and *Dependency* links which are *product-related*.
- Letelier et al. [13], also consider *satisfies* links with the *validatedBy*, *verifiedBy* and *assignedTo* relations. They also consider the *rationale* link type with the *rationaleOf* relation. Even more, they consider *modifies* and *reponsibleof* traceability relations which are relations among stakeholders (system's actors) and the different system specifications. Those relations are *process-oriented*.

As the analysis shows, authors have developed traceability relations to cover goal, task and resources dependencies and requirements satisfaction issues, which are *product-related* links. Another issue that is starting to be analyzed by several authors is product evolution and system development rationale such as [13, 15, 16].

[2] Product Family
[3] Feature Map
[4] Component Map

Evolution issue is, at present, partially covered by configuration management tools, which are more focused on system object history and system configuration history. There are some research initiatives oriented to capture the rationale information as in [4, 17, 18]. The objectives are to maintain the rationale that is behind a product, to justify the actions taken and why and how the product development items have been introduced. But the status of this kind of work can still be qualified as emerging.

2.2 Pre and Post-traceability issue

Some authors, such as Jarke [11] and Finkelstein et al. [8] are interested on pre-traceability issues, in order to provide the media to reopen and to rework previously closed issues and to management the requirements analysis process for producing a Requirement Specification, which really satisfies customer needs.

However many other authors are more interested on post-traceability issues, as it is shown by approaches such as, Jarke et al. [15], Spanoudakis et al. [16], Van Vliet et al. [12], Cerbah et al. [5], Egyed et al. [7], Macfarlane et al. [14], Watkins et al. [19].

2.3 Links Purpose

The traceability relation existence must be justified in a system development project. Each trace must have a reason to be a system artifact to analyze, to save, to monitor, and to present to the project decision makers.

Then, the following objectives have been identified:

- From the business view point, to provide the often demanded linkage between the business and IT [11], to manage the contributions made by the stakeholders and to manage the system development responsibilities assigned to them [13].
- From the technical view point, the purposes are to ensure the requirements are satisfied by the system, to assure that all requirement will be realized; to help managing dependencies among objects; to represent the rationale behind objects or documents; and to document actions leading from existing objects to new or modified objects (evolution) [13, 15, 16].
- For the Product Family issue, the aim is to define all the family features from its members; to show which and how many artifacts realize each product feature; and to show which and how many artifacts implement each design decision [12].

2.4 Linked Items Category

The items linked by the traceability relations presented by the authors of this analysis, belongs almost to the same category:

- System components in general: Design (class, interfaces, attribute, association, methods, etc), implementation (operations, program components, COTS, modules, documentation, etc) and requirement objects (requirement specification's requirements, formal user needs documents, etc) [8, 11–13, 15, 16].
- Test objects: Test plans, verification plans, test cases, test specifications, etc [8, 11, 13, 15].
- Resources: Physicals (money, electricity, etc) or informational (documents, data, etc) [8, 11, 13, 15].
- All pre-requirements stage documents: Requirement analysis process documents, approvals, meet minute, e-mail exchanges, decisions made, alternatives considered, underlying assumptions, stakeholder goals, etc [8,11,13,15].
- Stakeholders (system actors) [13].
- No textual and no model specifications: Videos, images, voice, etc [13].

Some authors also considered smaller granularity items to link such as, a requirement statement's sequence of terms [16], a use case's description [16], a requirement statement's description [16], etc.

2.5 Analysis

Authors have concentrated their studies on *satisfies* and *dependency* traceability relations which belongs to the *product-related* links, according to Jarke and Ramesh's [15] definition. Another issue identified is that evolution and rationale-based links issues are having an increasing interest by research and industrial sectors, as is indicated in [4, 13, 15–18].

In general, traceability link purposes, presented in literature, cover different stakeholders needs, from the more general such as, to enable safety analysis, audits, change control, support flexible process modeling, and compliance verification, to other more specific, such as technical or product family issue needs.

The type of items used by different authors are similar, though some of them analyze the possibility to link very specific parts of the items, such as a requirement statement's sequence of terms, use case's description, an use case's event, etc [12, 13, 16], in order to automate the link generation.

One of the problems identified is the lack of a standard scheme to define the link, so that it may be possible to set up support processes, such as to maintain the links, or other such as verification or validation, in a systematic way. The approaches do not specify which link's information must be stated for each traceability type, for example, to specifically define the allowed objects to link or the trace granularity, which are important issues to automate the trace's management. The approaches do not make recommendations about which traceability category is more feasible, useful and reliable to the several system types, for example, to embedded or computer-based systems or those based on MDA development.

Also, the approaches do not specify how to check for quality requirements, in order to assure an updated traceability strategy. Overlaps among definitions have been identified as well.

Then, as a conclusion it is necessary to define a Traceability Specification Scheme, the common features set detected from the traceability type's analysis will be helpful to define it. That scheme would specify the traceability classification necessary to the project, according to a standard guidance; also a link set which specifies all the needed links for a specific development step baseline, and a set of traceability metrics in order to verify some quality requirements such as, completeness, functionality, reliability, usability, efficiency, etc.

3 An initial proposal for a Traceability Scheme Specification

A Traceability Scheme Specification (TS) is an approach to specify the characteristics and needed items in order to be able to adequately support issues such management process, and facilitate maintainability, and project control. Some of the ideas could be in line with those underlying [6]. This issue will be further worked in the future.

The Traceability Scheme Specification should include the following items:

- A Traceability Link Dataset that will provide a wide basis to define traceability links, applicable a different kind of projects performed using different process models.
- A Traceability Link Type Set (TYS), in order to define which kind of information each traceability link type will contain.
- A Minimal Set of Traceability Links (MINS), in order to define all traceability link types, for a specific project or traceability baseline.
- A MEtriCs Set for the MINS (MECS), in order to define which metrics can or must be applied according to a measurement strategy, to verify if a correct traceability deployment and management is being performed.

3.1 Traceability Link Dataset

The concept of Traceability Link Dataset plays a similar role to that of metaclass in modeling. The idea behind is to provide a baseline to define all kind possible traceability links regardless the process used or the objective of the link. The dataset will include the following:

LINK TYPE The link type will have a type and will be related either to a software/system development product or development process.

DESCRIPTION It is a brief description of the traceability type.

PURPOSE Explain the reasons for the link type existence.

RELATED ITEMS Underlying the items or artifacts with which relate.

TYPE SUBCLASIFICATION For each link type it will be possible to define a number of sublinks with the following characteristics:

1. Link Subtype Name

   DESCRIPTION It is a brief description of the traceability subtype.

   PURPOSE Explain the reasons for the link subtype existence in a system development.

   RELATED ITEMS Underlying the artifacts with which it is possible to make a relation of this subtype.

   EXAMPLES Present link subtype examples

USES Present possible link type uses in a software project.

EXAMPLES It is presented some general examples of the link type.

3.2 Traceability Link Type Set

For a given project, process and product, it will be necessary to define those types of links that are felt as required. The template to be used will be that defined in the Traceability Link Dataset.

3.3 Minimal Set of Traceability Links

The Minimal Set of Traceability Links will define the set of links that should be created once a system baseline is closed. This set might be defined according to the stage of the development and the process.

The Traceability Specification should define the Minimal Set of Traceability Links (MINS). Once the life cycle steps are completed and a baseline of those steps are closed, it should specify the Life Cycle Step's Artifact Set (ARS), which contain all the system artifacts produced during those life cycle steps. Therefore, when each system baseline is established, there should be an ARS which includes all the system artifacts which that baseline includes.

The Minimal Set of Traceability Links (MINS) states the traceability links among the ARS's artifacts, produced during a specific baseline of a system life cycle. The links are created based on the traceability types defined on the Traceability Link Types Set (TYS). The MINS's link number depends on the traceability analysis that determines which and how many ARS's artifacts are needed to trace.

A traceability baseline should be stated when each system baseline does, and includes the MINS and ARS sets.

The MINS should include the following traceability links:

- For each ARS, the links among the artifacts themselves
- For all ARS, among an ARS set and its previous ARS (or vice versa the links among an ARS and the next ARS).

3.4 MEtriCs Set

This traceability scheme should be usable. After analyzing a number of commercial and non-commercial tools it was clear that usability was a key issue. Many tools implemented interesting features but usability was not one of the relevant features. Together with usability there some other characteristics that can be classified under the non-functional requirements umbrella and define the "quality" of a given scheme. To evaluate this level of quality some metrics should be defined to measure these no-functional requirements chosen for the traceability scheme.

The MEtriCs set will include all the traceability metrics to provide a basis that assure the correctness and level of accomplishment of the traceability strategy deployment.

For example, concerning integrity, the real traceability links number obtained during the measurement might be compared to the MINS number (the MINS set's cardinality) specified in the TS[5], in order to check for traceability completeness.

4 Conclusions and Future Work

This paper provides the result of a study aiming at identifying an existing traceability model to implement it into an existing software engineering environment. One of the objectives was to enhance the support of roundtrip engineering required to support verification, validation and non-conformance management activities. Available literature was studied and analyzed. The issues identified were lack of a standard scheme to define the traceability link; also the approaches do not make recommendations about which traceability category is more feasible, useful and reliable, for example, for embedded systems, computer based systems or systems based on MDA development. Even more, the approaches do not specify how to check quality requirements, in order to assure that an updated scheme is consistent. Overlaps among definitions have been identified as well.

As a result of the analysis, the need for a common traceability scheme was identified and a first approach for a Traceability Specification Scheme has been provided, whose objectives are to gather the common features of the traceability approaches analyzed and to provide a baseline to systematize and to improve the traceability control.

The scheme is made of a Traceability Types Set (TYS), which defines all trace types considered by the chosen traceability strategy; the Minimal Set of Traceability Links (MINS), which includes all traceability links for a specific traceability baseline; and the MEtriCs Set which defines the metrics to be applied by a measurement strategy. At present this scheme is being formalized. The possible relationship with specifications such as [6] are under study now.

[5] Traceability Specification

As future work, the Unified Traceability Scheme, will be implemented in the Software Engineering Environment (SEE[6]) [1], in order to improve the current SEE's traceability scheme.

It is expected that this Traceability scheme will be helpful to improve the management of the traceability and dependency relationships among MDA modeling elements as long as it intends to be able to integrate relevant features of existing traceability models.

Acknowledgement. This research work has been partially sponsored by the AGMOD project, Ref. TIC2003-08503, funded by the Ministry of Education of Spain. It is wanted to be grateful to the National Council of Science and Technology (CONACyT) of México for partially sponsoring this research.

References

1. P.P. Alarcon, J. Garbajosa, A. Crespo, and B. Magro. Automated integrated support for requirements-area and validation processes related to system development. In *Industrial Informatics, 2004. INDIN 04. 2004 2nd IEEE International Conference on*, pages 287–292. ISBN 0-7803-8513-6, 2004.
2. IEEE Standards Board. *IEEE Std 610.12-1990 IEEE standard glossary of software engineering terminology*. IEEE, Institute of Electrical and Electronics Engineers, 345 East 47th Street, New York, NY, September 28, 1990.
3. Alan W. Brown. Model driven architecture: Principles and practice. *Software and System Modeling*, 3(4):314–327, 2004.
4. J. Burge and D. C. Brown. Reasoning with design rationale. In *Artificial Intelligence in Design 00*, pages 611–629, Netherlands, 2000. Kluwer Academic Publishers.
5. Farid Cerbah and Jérôme Euzenat. Using terminology extraction to improve traceability from formal models to textual requirements. *Lecture Notes in Computer Science*, 1959:115–126, 2001/01. http://www.springerlink.com/index/8T69TBYUCJ994T1U.
6. ECMA. *ECMA-149: Portable Common Tool Environment (PCTE) — Abstract Specification*. pub-ECMA, pub-ECMA:adr, fourth edition, dec 1997.
7. A. Egyed and P. Grunbacher. Automating requirements traceability: Beyond the record and replay paradigm. In *Automated Software Engineering, 2002. Proceedings. ASE 2002. 17th IEEE International Conference on*, pages 163– 171. IEEE, 2002.
8. O.C.Z. Gotel and C.W. Finkelstein. An analysis of the requirements traceability problem. In *Requirements Engineering, 1994., Proceedings of the First International Conference on*, pages 94–102, Colorado Springs, Co., 18-22 Apr 1994. IEEE Computer Society Press.
9. ISO – International Standard Organization. *(ISO/IEC 12207) Information Technology—Software Lifecycle Processes*, 1995. 85 S.
10. ISO – International Standard Organization. *(ISO/IEC 15288) Systems Engineering—System life cycle processes*, 2002.
11. Matthias Jarke. Requirements tracing. *Commun. ACM*, 41(12):32–36, 1998.

[6] Fully compliant with ECSS-E40 standard process model for developing on-board embedded real-time software

12. P. Lago, E. Niemelä, and H. Van Vliet. Tool support for traceable product evolution. In *CSMR '04: Proceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering (CSMR'04)*, pages 261–269, Washington, DC, USA, 2004. IEEE Computer Society.
13. Patricio Letelier. A framework for requirements traceability in UML-based projects. In *1st International Workshop on Traceability in Emerging Forms of Software Engineering. (In conjunction with the 17th IEEE International Conference on Automated Software Engineering)*, pages 173–183, Edinburgh, U.K., September 2002.
14. I.A. Macfarlane and I. Reilly. Requirements traceability in an integrated development environment. In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on*, pages 116–123. IEEE Computer Society, 27-29 Mar 1995.
15. B. Ramesh and M. Jarke. Toward reference models for requirements traceability. *Software Engineering, IEEE Transactions on*, 27(1):58–93, Jan 2001.
16. George Spanoudakis, Andrea Zisman, Elena Pérez-Miñana, and Paul Krause. Rule-based generation of requirements traceability relations. *Journal of Systems and Software*, 72(2):105–127, July 2004.
17. Saïd Tazi and David G. Novick. Design rationale for complex system documentation. In *Proceedings of the Conference on Complex Systems, Intelligent Systems and Interfaces (Nimes 98)*, volume combined volumes 134-236, pages 49–51, Nimes, France, 1998.
18. Arie Van Deursen. Recovering rationale. In *Proceedings of the WCRE 2001 Discussion forum on Software Architecture Recovery and Modeling (SWARM 2001)*, pages 9–10, Netherlands, 2001. CWI.
19. R. Watkins and M. Neal. Why and how of requirements tracing. *Software, IEEE*, 11:104–106, Jul 1994.