A Criteria-Based Approach to Classifying Traceability Solutions

Petro Protsyk
International Software & Productivity Engineering Institute
protsyk@intspei.com

Konstantin Zhereb
International Software & Productivity Engineering Institute
kzhereb@intspei.com

Abstract

The goal of traceability is to establish and trace relationships between different project artifacts. Traceability provides understanding of the dependencies that exist among artifacts and allows performing impact analysis, generating test cases, etc. In the paper, we propose a new method for the classification of traceability solutions based on a developed numeric scale. This scale was used to evaluate the quality rate of the different existing traceability approaches. As a result of our evaluation, we found the new INTSPEI P-Modeling Framework to be the most advanced traceability solution.

1. Introduction

Traceability [1] is recognized as an important activity during the project development cycle. Its goal is to establish and track different relationships between project artifacts, in both forward and backward directions.

Traceability relationships help stakeholders understand the dependencies that exist among artifacts. The dependencies in turn allow the stakeholders to perform progress and impact analysis, i.e. to estimate and reduce the amount of efforts needed to make a change to an artifact, its dependent artifacts and then to test them. However, the main purpose of traceability is to ensure that correct outcomes are produced at each step of the project development life cycle.

The key factors in traceability are the relationships between artifacts and their tracking. Complex projects contain a large number of artifacts of different natures and origins. Therefore, there may also be various types of relationships, and these various types of relationships are traced differently. Unfortunately, there is no uniform approach for the classification of these relations. Moreover, there are no automated procedures for estimating the quality of the relationships between different artifacts, usually they are processed manually, i.e. by a human. These, and many other aspects, make maintaining traceability into an existing project development process a difficult task.

In the paper, we discuss existing classifications of traceability approaches and present a new classification according to three main criteria: traceability information, methodology and tool support. Based on these criteria, we propose a numeric scale for evaluating the sophistication of the traceability solutions.

We also describe and analyze the INTSPEI P-Modeling Framework approach. The framework is based on two key principles: Reverse Semantic Traceability and Speechless Modeling. Reverse Semantic Traceability is a new method of maintaining and conducting traceability of project artifacts. The essence of the method is the way in which the quality of the relationships between artifacts is estimated. This is called the Reverse Translation step: before proceeding to the next phase, the current output artifacts are "reverse engineered" by human engineers, and these restored artifacts are semantically compared to the original input artifacts. If there is a significant difference between them, they have to be corrected to eliminate the problem. Semantic comparison within this context means that the meanings of the artifacts have to be compared, not the exact words.

The rest of the paper is organized as follows: We present a classification of existing traceability approaches together with Reverse Semantic Traceability. Based on this classification, we evaluate the quality and sophistication of the existing traceability solutions. Then we concentrate on the particular benefits of the P-Modeling Framework approach and make our conclusions.

2. Classification of traceability approaches

Some classifications of traceability approaches are currently present in literature. However, most of them concentrate only on particular details of traceability

problems, basically ignoring the overall picture. In this chapter, we discuss the proposed classifications and establish classification schemes, generalizing the existing approaches.

Aizenbud-Reshef and others [2] discuss traceability models and their connection to the model driven development (MDD) approach. Traceability solutions are discussed based on five criteria: metamodel (what types of artifacts and links are supported); linkage information (how links are stored); automated creation (how link creation is automated); relationship management (how links can be updated based on changes in artifacts); and methodology (how the traceability process integrates into existing development cycle). This classification captures many aspects of traceability problems. However, we believe that some important questions are omitted from this classification. In particular, important information about why traceability processes are used and what roles are involved in these processes is not included. These questions are mentioned by Aizenbud-Reshef and others [2], but their overview is spread between the metamodel and methodology sections. We believe that these questions deserve their own separate section. Also, some automation aspects are missing from this paper [2]: automated creation and change management of links are discussed, but deriving information about the state of the project or particular artifacts is interwoven with change management (as a computation of some links based on the values of others). We propose distinguishing between change management and retrieval of information based on links. These activities are performed on different levels and by different stakeholders. Furthermore, not all information about traceability should be presented as special types of links.

Some classification approaches concentrate on the metamodel aspect of traceability. In their paper, Ramesh and Jarke [3] present extensive research of traceability usage in the industry. Based on this research, the authors propose a generic metamodel for traceability. Three core classes are identified: Object – logical artifacts that can be traced; Stakeholder – users who are involved in the traceability process; and Source – physical representations of artifacts and traceability links. Ramesh and Jarke [3] also provide classification of traceability links as product-related (satisfied and dependency), or process-related (rationale and evolves-to). Another question this paper [3] discusses is the distinction between two possible ways of using traceability: high-end and low-end traceability. This distinction concerns both categories of users that participate in the traceability process, and reasons for using traceability. Unfortunately, this paper [3] omits the questions related to traceability automation.

There are also partial classifications of traceability models. Czarnecki, and Helsen [4] provide traceability classification as a part of the classification of model transformation approaches. Traceability schemes are distinguished by two major criteria: storage location (where links are stored) and control (how they are created). Cleland-Huang [5] evaluates the degree of automation in traceability approaches based on three elements: trace creation & maintenance; identification of impacted artifacts; and evaluation of the scope of the impact. This classification is tailored for impact analysis applications. Gotel and Finkelstein [6] identifies the distinction between pre-RS (Requirements Specification) and post-RS traceability problems.

Based on the analysis of the existing traceability classification schemes, we propose a new classification that generalizes current approaches. Our classification approach is based on the following criteria:

- Traceability information (what artifacts can be connected by links; what attributes are associated with links; what metamodel is used; and how traceability information is stored);
- Methodology (why traceability is used; what categories of users participate in the traceability process; and how traceability is integrated into an existing SDLC);
- Tool support (degree of automation in creating links; updating links; and deriving information from links);

We believe that this classification scheme concentrates on the important aspects of traceability. It provides three interconnected views on traceability problems. Traceability information defines the essence of a particular traceability approach: what information about traceability links is considered? The methodology concentrates on how people use traceability. Finally, automation of the traceability process can be described as tool support. Furthermore, our classification scheme is a superset of current classification approaches.

Based on our classification, we develop numeric scales to measure the sophistication of traceability solutions. For each traceability approach, we want to evaluate two numeric parameters: sophistication of methodology (Me) and the quality of tool support (TS). These parameters represent methodology and tool support criteria respectively. Traceability information criterion does not correspond to a numeric parameter. This distinction is not made because of a lesser

importance of linkage information compared to other criteria. Features related to traceability information are crucial for traceability solutions, but they also constitute inner details of a particular traceability approach. On the other hand, the sophistication of methodology and quality of tools are properties that describe high-level aspects of any traceability solution; furthermore, they depend on traceability information.

To calculate values of Me and TS, we first establish a list of features grouped by major criteria from our classification: traceability information, methodology and tool support. With each feature, we associate its weight (number from 0 to 10) within Me and TS metrics. Most features from methodology and tool support groups affect only the corresponding value (Me and TS, respectively). Traceability information features are different in that they do not have associated numeric parameters. Instead, they affect both Me and TS with different weights.

After the list of features has been established, we can rate each traceability solution according to this list. For each feature, we assign a rate of 0 (if the approach in question does not implement the feature) or a number from 1 to 5, according to the quality of implementation. To obtain values of Me and TS, we calculate the weighted sum of rates of all features.

In the following sections, we will establish a list of traceability features according to our new classification scheme. Next, we will calculate the values of parameters for existing traceability approaches, as well as the INTSPEI P-Modeling Framework.

2.1. Traceability information

First, we will list features of traceability solutions according to information associated with traceability links. The features are listed in the following format: name; weights for Me and TS values; description. The following properties are important:

1. What artifacts can be connected by links (Me=7, TS=1)? This describes what types of artifacts can participate in traceability relationships. A high rate implies that a traceability link can exist between any artifacts.
2. Type of link present (Me=9, TS=1). This describes whether links have distinct types that describe their purpose. A high rate implies a complex type system.
3. Reasoning based on link types (Me=3, TS=9). This describes whether or not the system can perform actions based on types of links.
4. Support for unidirectional/bidirectional links (Me=3, TS=0). This describes support for various directions of links. A simple approach may support only links from a high-level artifact to its implementation, while more complex solutions will include various kinds of directions.
5. Support for 1-n, n-n links (Me=10, TS=8). This describes support for complex links, establishing a relationship between more than 2 artifacts.
6. Support for numeric attributes (Me=8, TS=7). This describes whether or not the system supports numeric values associated with the link (link quality, complexity, etc.).
7. Support for user-defined attributes (Me=5, TS=8). This describes whether or not the system allows a user to define new attributes associated with links.
8. Retrieval of data based on attributes (Me=2, TS=10). This describes if it is possible to construct reports according to values of link attributes.
9. Linkage information can be stored in separate storage (Me=1, TS=8). This describes whether or not links can be stored in a separate storage from artifacts. This parameter affects the possibility of integration with external tools.
10. Storage options can be adjusted (Me=1, TS=5).
11. Complexity of traceability metamodel (Me=5, TS=1). This is a general criterion, describing the sophistication of the metamodel used in the traceability process.

2.2. Methodology

In our classification, the methodology section concentrates on how people use traceability solutions. The most important considerations are:

1. Involvement of various roles (project manager, requirement manager, designer, developer, tester) in the traceability process (Me=8, TS=0). Rated separately for each role.
2. Used to link requirements to final components (Me=5, TS=0).
3. Used to understand links between artifacts in arbitrary stages of development cycle (Me=9, TS=0). This differs from previous aspect in that it assigns meaning to links between artifacts on different abstraction levels, not just between code and requirements.
4. Used to obtain coverage information (Me=10, TS=0). This describes how a good traceability solution answers questions about the coverage of requirements by code, requirements by tests, code by tests, etc.
5. Used to evaluate status of an individual artifact (Me=6, TS=0). Describes how traceability can estimate an artifact based on links to it.

6. Used for impact analysis (Me=10, TS=0). This describes how traceability can estimate the impact of changes in requirements.
7. Established procedures for assigning links/link attributes (Me=10, TS=0). This describes whether or not the traceability solution describes procedures for establishing links between artifacts and assigning attributes to links.
8. Integration with popular methodologies (RUP, MSF, XP) (Me=10,TS=0). Evaluated separately for each methodology.
9. Can integrate into new methodologies (Me=7, TS=0).

2.3. Tool support

Automation of the traceability process is very important because of the large number of artifacts and links involved. Important aspects of automation include:

1. Visualizing traceability state (Me=0, TS=8). This describes how a system can present traceability information in visual form.
2. Project-level reports (Me=0, TS=10). This describes what reports can be produced about the overall state of project.
3. Artifact-level reports (Me=0, TS=8). This describes what reports can be produced about individual artifacts or groups of artifacts.
4. Automatic link creation (Me=0, TS=7). This describes whether or not the system can automatically create links between artifacts based on their content.
5. Adjusting options for automatic link creation (Me=0, TS=5). This describes whether or not a user can adjust options for link generation (e.g. to balance precision – number of wrong links created and recall – number of correct links missed).
6. Creation of links as part of workflow (Me=0, TS=10). This describes whether or not the system can automatically create links when an action is performed (new artifact generated, new link added, etc.).
7. Support for changing links (Me=0, TS=7). This describes how a solution manages changes of links resulting from changing artifacts.
8. Integration with file system, Internet resources, other common storage systems (Me=0, TS=10).
9. Integration with project management, version control and other specialized development tools (Me=0, TS=10).
10. Integration with common file formats (plain text, XML, HTML, Word, Excel, PDF) (Me=0, TS=10).

3. Evaluation of traceability approaches

We have evaluated some traceability approaches according to our developed scales – methodology and tool support. A review of current traceability solutions was performed, and we concentrated on the differences between the existing approaches, in terms of three major criteria: traceability information, methodology and tool support.

In the field of traceability information, the main difference lies in the complexity of the metamodel being used. Most solutions employ a very simple metamodel: a single type of traceability link and little distinction between artifact types. However, some research projects (in particular, Ramesh and Jarke [3], Espinoza, Alarcon, and Garbajosa [7] and Letelier [8]) concentrate on creating advanced metamodels for the traceability process. Another important distinction is support for specialized attributes of traceability relations. Such attributes are rarely used because there are no established procedures for evaluating traceability links.

When methodology is considered, the main question is how the traceability process integrates into existing software development processes. We found that many solutions provide integration with more formal methodologies (like RUP and MSF), but support for agile processes is not very common. There is also a distinction between reasons for using traceability, discussed by Ramesh and Jarke[3], as low-end and high-end uses. While the former view demonstrates traceability as something important only for customers, the latter fully utilizes traceability in all aspects of development.

As for tool support, most solutions provide some way of visualizing traceability information or producing reports. However, there is little automation of more complex aspects of traceability. In particular, only research projects [2, 11] provide some means of automated link creation. There are various degrees of integration with external applications: most commercial tools can read at least most common file formats (HTML or Microsoft Word), but full integration with development tools is rare.

Based on our review, we selected several traceability solutions that represent the spectrum of traceability approaches. The simplest approach is using a traceability matrix [2] (marked as TM) to store information about the evolution of each artifact. This

solution provides a basic level of methodology and very little tool support. However, it is widely used because of its simplicity. More complex approaches are either research projects or commercially available tools.

Many research approaches make extensive use of metamodels – low-end and high-end usages of traceability, as identified by Ramesh and Jarke [3], as well as traceability framework, proposed by Letelier [8]. The difference between these approaches is the complexity of the metamodel they use. Low-end users are primarily concerned with establishing links between initial requirements and final components. They use traceability to find components implementing the given requirement or requirements affecting a particular component. High-end users have a wider spectrum of uses for traceability – requirements management, rationale management, design allocation, and compliance verification. Finally, the traceability framework [8] is an attempt to construct a generic metamodel that can be easily tailored to a specific project.

Antoniol and others [11] offer a system using information retrieval (IR) techniques as an example of an approach that concentrates on the automation of the traceability process. The authors use statistical methods that calculate linkage information from the coincidence of terms used in artifacts. Aizenbud-Reshef and others [2] offer another approach to automated link generation according to the statistics of changes. This approach analyzes change history and suggests creating links for groups of artifacts that have been changed together frequently. Other similar methods have been introduced by Aizenbud-Reshef and others [2] and Cleland-Huang [5]. Such methods support very basic methodology, concentrating on tool support aspects instead.

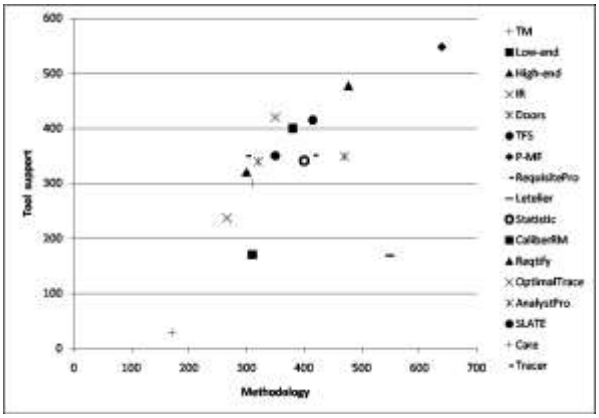

Figure 1. Evaluation of traceability solutions

As opposed to research projects, we have evaluated some commercially available products. We considered several examples of specialized tools for traceability management: Telelogic Doors [9], IBM Rational RequisitePro [10], Borland Caliber RM, Compuware Optimal Trace, ChiasTek Reqtify, etc [15]. We also used Microsoft Team Foundation Server/Visual Studio Team System solution as an example of general development tools that can also be used to support traceability.

Finally, we evaluated the INTSPEI P-Modeling Framework (P-MF) [12-14]. This approach provides support for numeric attributes associated with links describing the quality of the relationship between artifacts. These attributes are computed as a result of Reverse Semantic Traceability and can be used to evaluate the quality of transition between artifacts. P-Modeling Framework emphasizes the ability to integrate with existing SE methodologies (RUP, MSF, and XP). In the field of automation, INTSPEI P-Explorer (traceability tool supporting P-MF [14]) provides visualization and reports based on the presence of links between artifacts, as well as a numeric attribute assigned to these links. Integration with various artifact storage systems is also provided. INTSPEI P-Modeling Framework does not provide custom support for automated link creation; however, it can rely on other methods to create such links.

Solution	Methodology	Tool Support
Borland Caliber RM	340	400
ChiasTek Reqtify	300	320
Compuware Optimal Trace	350	420
Goda Analyst Pro	320	340
High-end	480	480
IBM Rational RequisitePro	400	400
Information Retrieval	280	230
Letelier	550	180
Low-end	320	180
P-Modeling Framework	650	550
SLATE Require	350	350
Sophist Care	310	380
Statistics	400	350
Team Foundation Server	450	450
Telelogic Doors	480	350
Traceability Matrix	180	50
XTie Requirements Tracer	350	350

Table 1. Solutions summary

The combined results of evaluating traceability solutions are presented in Figure 1 and in Table 1. Two parameters – methodology and tool support – are shown. The figure shows that there are four clusters of approaches divided by sophistication. The most basic solution for traceability is using a traceability matrix. The next cluster is low-end users according to Ramesh

and Jarke [3], as well as IR techniques. High-end users, as well as commercially available tools, constitute the third group. Finally, the INTSPEI P-Modeling Framework belongs to the fourth cluster.

4. Solutions Overview

Here we present an overview of some of the traceability solutions we considered. The overview is based on the public information provided by manufacturers through their web sites.

Borland CaliberRM is a requirements management tool with some traceability support. Some of the key features include: centralized repository for project artifacts, customization tools to adopt the solution for an existing process, requirements traceability across the application lifecycle, real time impact analysis, and a diverse set of clients for Web, Eclipse, MS Visual Studio. CaliberRM has an open architecture that permits requirements to be linked to a variety of artifacts across the lifecycle. It also has multiple methods of traceability visualization to help users immediately understand the scope of analysis required to gauge the effect of a requirements change.

Optimal Trace is Compuware's business requirements definition and management tool. The approach associated with the tool offers a standard structure for all projects and corresponding documents and reports. The traceability functionality of Optimal Trace ensures full visibility of the connections and interdependencies within and across projects and between elements of the project and external file artifacts. The tool uses an automatic heuristics algorithm for predicting changes that may have a ripple effect on other project requirements.

XTie Requirements Tracer is a requirements management and analysis tool that resulted from 10 years of internal development. It was developed to ensure product quality and integrity by managing the requirements and enforcing the traceability of requirements throughout a systems development life cycle. It is able to trace requirements from design and development through completion of the project. Requirements Tracer contains a database manager designed specifically to support the cataloging and manipulation of information across sets of specifications. It features fast access to structured requirements data and exceptional support for the requirements analysis process.

INTSPEI P-Modeling Framework originates from "The Babel Experiment" designed by Vladimir L. Pavlov in 2001 as a training program [16]. It is based on two powerful principles: Speechless Modeling and Reverse Semantic Traceability. Reverse Semantic Traceability can be considered as the quality control method that allows testing outputs of every "translation" step. Before proceeding to the next phase, the current outputs have to be restored and compared to the original artifacts. If the artifacts are semantically different, then the translation steps have to be repeated more precisely to eliminate this ambiguous understanding. This reverse step and the degree of semantic correctness must be performed by the engineers.

5. Conclusions

In conclusion, our article presents an approach to the classification of traceability solutions. It is based on three major criteria: traceability information, methodology and tool support. These criteria are chosen as a generalization of currently existing approaches. Based on this classification, we have proposed the numeric scale for evaluating quality and sophistication of traceability solutions.

Additionally, we evaluated and discussed current solutions together with the new INTSPEI P-Modeling Framework, establishing the positions of those solutions within the proposed classification scheme. Our research shows that existing traceability solutions cannot efficiently detect flaws introduced in the early stages of a development process instantly. High-level artifacts can be traced to low-level implementations and tests, but they cannot be verified until these low-level artifacts are created. The INTSPEI P-Modeling approach allows verifying transitions made at each step of development cycle.

Future research will include developing methods for calculating various project metrics based on values of numeric attributes of traceability links. The types of possible attributes that can be applied to traceability links is also a matter of individual research.

6. Acknowledgments

We would like to thank all people who reviewed different versions of this article, especially: Vladimir L Pavlov, Eric Gilliatt, Anatoliy Doroshenko and Bekah Sondregger.

7. References

[1] Donald Firesmith, Specifying Good Requirements, Journal of Object Technology, Vol. 2, No. 4, July-August 2003.

[2] N. Aizenbud-Reshef, B. T. Nolan, J. Rubin, Y. Shaham-Gafni, Model traceability. IBM SYSTEMS JOURNAL. Vol. 45, No. 3, 2006.
[3] B. Ramesh and M. Jarke, ‘‘Toward Reference Models for Requirements Traceability,’’ IEEE Transactions on Software Engineering 27, No. 1, 58–93 (January 2001).
[4] K. Czarnecki, S. Helsen; “Classification of Model Transformation Approaches”. Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, USA, 2003
[5] J. Cleland-Huang, “Toward improved traceability of non-functional requirements”. Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering, California, USA, 2005
[6] O. C. Z. Gotel and A. C. W. Finkelstein, ‘‘An Analysis of the Requirements Traceability Problem,’’ Proceedings of the First International Conference on Requirements Engineering, Utrecht, The Netherlands (1994), pp. 94–101.
[7] A. Espinoza, P. P. Alarcon, J. Garbajosa, "Analyzing and Systematizing Current Traceability Schemas", pp. 21-32, 30th Annual IEEE/NASA Software Engineering Workshop SEW-30 (SEW'06), 2006.
[8] P. Letelier, "A Framework for Requirements Traceability in UML-Based Projects", Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering, Edinburgh, Scotland (September 2002), pp. 30–41.
[9] DOORS, Telelogic, http://www.telelogic.com/ products/doorsers/index.cfm.
[10] RequisitePro, IBM, http://www-306.ibm.com/software/ awdtools/reqpro/
[11] G. Antoniol, G. Canfora, G. Casazza, A. D. Lucia, and E. Merio. Recovering traceability links between code and documentation. IEEE Transactions on Software Engineering, 28(10), October 2002.
[12] Pavlov, Vladimir; Yatsenko, Anton “Using Pantomime in Teaching OOA&OOD with UML”, *18th IEEE Conference on Software Engineering Education and Training*, April 18, 2005.
[13] Vladimir L. Pavlov, Nikita I. Boyko, Alexander V. Babich, “First Experience of Using INTSPEI P Modeling Framework in Software Development projects”, Problems In Programming, Issue 2, 2007, pp. 68–75.
[14] INTSPEI P-Modeling Framework Whitepaper, INTSPEI, http://www.intspei.com
[15] INCOSE Requirements Management Tools Survey, http://www.paper-review.com/tools/rms/read.php
[16] Vladimir L Pavlov, Stanislav Busygin, Nikita Boyko, Alexander Babich, “Is There Still a Room For Programmers' Productivity Improvement?”, to appear in Proceedings of the 5th East-West Design and Test Symposium (EWDTS'07), 2007