# A Framework for Traceability Tool Comparison

Patrick Rempel, Steffen Lehnert, Tobias Kuschke, Qurat-Ul-Ann Farooq

Ilmenau University of Technology
Ilmenau, Germany
{patrick.rempel, steffen.lehnert, tobias.kuschke, qurat-ul-ann.farooq}@tu-ilmenau.de

## Abstract

The need for automated traceability support has been advocated by various researchers and practitioners alike, as manual traceability is cumbersome and error-prone. Thus, many tools and approaches have been developed to overcome the limitations of manual traceability detection and usage. However, the large amount of available traceability tools is lacking suitable means for reasonable comparison based on well-defined criteria, which in turn complicates the process of choosing the appropriate tool. We propose and discuss a framework for classifying traceability tools, demonstrate its feasibility by organizing three tools according to its criteria, and illustrate how its criteria can be used when searching for a suitable tool.

## 1 Introduction

The concept of traceability supports many different software engineering activities, including impact analysis [10], reverse engineering [12], requirements tracing [8], software maintenance [1], and consistency checking [13]. However, manual creation and maintenance of traceability relations is a time consuming and error-prone task [6], which instead should be performed by automated tools.

This need has been recognized by researchers and industrial partners alike, and many automated or semi-automated approaches for traceability detection, maintenance, and utilization have been developed. In 2011, Gae et al. [4] evaluated requirement traceability tools by surveying their vendors. In preparation for this vendor survey, the authors assembled a list of relevant requirements traceability tools from following tool databases: *INCOSE*,[1] *Volere*,[2] *QAGuild*,[3] *Ian Alexander*,[4] and three others. After discarding invalid and duplicate tools, Gae et al. identified no less than 94 vendors, which illustrates the impressive amount and diversity of traceability tools.

However, after many years of research and development, and the large amount of tools and approaches available, it is hard to select the right tool for a certain problem or task. When multiple tools share common features and capabilities, it is crucial to be able to refer to a set of well-defined criteria for selecting the most suitable tool. While related work recognized the need for comparing the performance of traceability approaches using traceability benchmarks [3], not much work has been spent on defining comparison criteria for traceability tools easing the actual tool selection.

In this paper we propose a framework which supports the following three activities (A1-A3):

**A1.** Enable the classification of tools.
**A2.** Enable the comparison of classified tools.
**A3.** Enable stakeholders to search for tools.

The contribution of our work is therefore twofold. First, we introduce a framework for classifying and comparing traceability tools. The criteria of our proposed framework are based on analysis of related work regarding traceability benchmarks and tool comparison, as well as analysis of structured CASE tool characteristics as standardized by ISO/IEC. We further evaluate and refine our criteria by analyzing five different types of stakeholders, which are typical users of traceability tools. Second, we demonstrate the applicability and usefulness of our criteria. We achieve this by classifying three existing traceability tools (A1) and illustrate by example how it can be used to search for traceability tools (A3). The comparison of tools (A2) is not covered in this paper due to space constraints.

The remainder of this paper is organized as follows. Section 2 derives requirements for traceability classification criteria. We then introduce our criteria in Section 3 and evaluate them in Section 4.

## 2 Towards Classification Criteria

The main goal of this section is to identify and specify requirements for traceability tools, which we accomplish be studying three different data sources. First, we study relevant traceability literature, we then infer traceability capability requirements from an existing ISO/IEC standard, and finally derive additional requirements by analyzing stakeholder specific needs. Finally, we summarize all identified requirements in a

[1] http://incose.org/ProductsPubs/products/rmsurvey.aspx
[2] http://www.volere.co.uk/tools.htm
[3] http://www.qaguild.com/toolsdirectory.aspx
[4] http://easyweb.easynet.co.uk/~iany/other/vendors.htm

comprehensive list, which serves as a baseline to derive our criteria in Section 3.

### 2.1 Related Work

The following discusses related work which is focused on establishing classifications for requirements engineering tools, architectural knowledge management tools, and traceability benchmarks. Related work on classifying requirements tools and architectural knowledge management tools is studied because they are closely related to traceability and thus share similar criteria. We analyze traceability benchmarks as they provide criteria for distinguishing and comparing different traceability approaches based on their performance, which is also important for comparing tools.

The traceability community identified the need for establishing traceability benchmarks to be able to properly evaluate and compare traceability approaches [3]. In their position paper, Dekhtyar et al. [5] discuss the need and organization of a possible traceability benchmark. The authors investigate the use of traceability for software validation, maintenance, and reverse engineering. Based on this investigation, Dekhtyar et al. derive their requirements for a traceability benchmark and propose the usage of the following criteria: *accuracy* (i.e. *precision*), *coverage* (i.e. *recall*), *scalability* (i.e. *computation time*), and *effort* (i.e. *manual effort*).

Charrada et al. [2] developed a benchmark for traceability which is based on the following three criteria, which are identical to those proposed by Dekhtyar et al. [5]: *precision*, *recall*, and *time*. The evaluation of their benchmark was accomplished by analyzing one traceability mining tool and gauging its performance according to the proposed criteria.

The following discusses two approaches for comparing different kinds of tools which both are closely related to traceability: requirements engineering and architectural knowledge management.

Similar to the need for comparing traceability tools, requirements engineering (RE) tools also demand for proper classification criteria to ease the selection of tools and to allow for comparisons of different tools. Gea et al. [4] compare RE tools according to the features offered by the tools, which is also reflected by the choice of their criteria, as eight out of ten represent typical applications of RE tools. The authors propose the following criteria: *Elicitation*, *Analysis*, *Specification*, *Modeling*, *Verification & Validation*, *Management*, *Traceability*, *Other tool capabilities*, *Global* (i.e. whether tools cover all criteria), and *Price range*.

A study comparing five architectural knowledge (AK) management tools was conducted by Tang et al. [14]. The authors defined a set of ten criteria to answer their two main questions: how tools can be embedded into existing development environments or processes, and which activities are supported by the tools. Tang et al. define the following criteria: *supported types of AK*, *relations between AK*, *arch. analysis support*, *arch. synthesis support*, *arch. evaluation support*, *arch. implementation support*, *arch. maintenance support*, *personalization of tool*, *integration with other tools*, and *multi-user support*.

In summary, the following requirements were identified by our literature analysis:
**RR1:** Purpose and provided features [14, 4].
**RR2:** Integration with other tools [14].
**RR3:** Performance [5, 2].
**RR4:** Usability and data visualization [14].
**RR5:** Standards compliance [14, 4].
**RR6:** Data definition and storage [14].
**RR7:** Multi-user support [14].

### 2.2 ISO/IEC Standard Analysis

In this section, we derive traceability related requirements from a framework for evaluating RE tool capabilities (ISO/IEC TR 24766:2009[5]). The ISO/IEC TR 24766:2009 is a technical report that supplements the ISO/IEC 14102:2008 standard, which defines a structured set of CASE tool characteristics. ISO/IEC TR 24766:2009 lists a total of 157 RE tool capabilities, which can be clustered into six major categories: *requirements elicitation*, *requirements analysis*, *requirements specification*, *requirements verification and validation*, *requirements management* and *other capabilities*. The category *other capabilities* contains the following capabilities: *traceability*, *flexible tracing*, *bidirectional tracing*, and *traceability analysis*. These traceability related capabilities can be further detailed to traceability requirements for RE tools.

From the complete list of traceability related requirements, we consider the following requirements as most important:
**RS1:** Manage traceability lifecycle.
**RS2:** Integrate with other tools.
**RS3:** Comply with standards.
**RS4:** Usability and data visualization.
**RS5:** Data definition and storage.
**RS6:** Traceability automation.
**RS7:** Traceability analysis.

### 2.3 Stakeholder Analysis

When designing a framework for traceability tool comparison, it is important to address the needs of different types of stakeholders. The *Center of Excellence for Software Traceability*[6] defines several users of their proposed *TraceLab* infrastructure for traceability benchmarking [3], which we adapt and extend for our framework for traceability tool comparison.

- Researcher: she is interested in comparing the performance of traceability tools, and requires interoperability with other tools to be able to conduct her case studies.

[5] http://www.iso.org/
[6] http://www.coest.org/

- Analyst: a requirements analyst utilizes pre- & post-requirements traceability to trace the origin and implementation of requirements. When selecting a tool, her focus is on the support for various requirements standards, and the ability to trace the lifecycle of requirements.
- Architect: the architect of a software system uses traceability to verify and validate the architecture against the requirements of a project. She further utilizes traceability for impact analysis and to assess the overall coupling between architectural components.
- Developer: she requires assistance for program comprehension, feature localization, impact analysis, and test selection. The interoperability with other tools and the support for various data types are of importance to her.
- Project Manager: she utilizes traceability to assess the current state of a project, e.g. by creating status reports, and to plan further development steps.

Based on the needs of these five stakeholders we can derive further requirements for the criteria of our framework.
**RSH1:** Support stakeholder specific activities.
**RSH2:** Support different artifact types.
**RSH3:** Provide interoperability with other tools.
**RSH4:** Provide information about dependent tools.

### 2.4 Unification of Requirements

In this section we unify the previously identified requirements to provide a baseline for our classification criteria. The following list shows the result of the unification procedure:
**R1:** Purpose and provided features. (RR1, RSH1)
**R2:** Integration with other tools. (RR2, RSH3, RS2)
**R3:** Traceability analysis.
**R4:** Usability and data visualization. (RS4,RR4)
**R5:** Standards compliance. (RR5,RS3)
**R6:** Data definition and storage. (RR6, RS5)
**R7:** Multi-user support.
**R8:** Support different types of artifacts.
**R9:** Provide information about dependent tools.
**R10:** Manage and automate traceability lifecycle.
**R11:** Performance.

## 3 Defining Classification Criteria

In the following section, we introduce and explain the criteria of our framework, which is aimed at supporting the classification (A1), comparison (A2), and search for traceability tools (A3). According to conclusions drawn in Section 2.4, we consider the following criteria as relevant to accomplish the activities A1-A3 and to fulfill our requirements R1-R10. Please note that R11 is not covered by our criteria because "Performance" cannot be generalized (e.g. precision and recall of a link detection tool cannot be compared to precision and recall of a link maintenance tool). These issues are extensively discussed in related literature on traceability benchmarks, as discussed in Section 2.1. We like to point out that the order in which we present our criteria does not reflect any ranking according to their importance.

**(C1) Main Application:** describes the main purpose and core features of a tool, e.g. impact analysis, feature localization or requirements traceability.

**(C2) Interoperability:** reflects how a tool adheres to standards, file formats, is able to interact with other tools, and supports multiple users. Thus *Interoperability* features the following sub-criteria: *Supported Formats*, *Supported Standards*, *Supported Tools*, and *Multi-user Support*. The support for multiple data formats or standards is especially important since no tool is being used in isolation. Likewise, multiple developers are typically involved and are working on the same set of data which is why multi-user support is required.

**(C3) Supported Artifacts:** provides an overview about all types of artifacts supported by a tool. It is further refined into: *Requirements*, *Architecture*, *Source Code*, *Test Cases*, and *Others*. This criterion is a reliable indicator whether a certain tool supports horizontal traceability or not (i.e. it is able to link different types of artifacts [9]).

The following two criteria (*Link-Creation* and *Link-Maintenance*) are very essential features which should be provided by any traceability tool. Thus we list them separately and refine them with further sub-criteria.

**(C4) Link-Creation:** is used to determine how traceability relations are created, which approach is used, and if it supports automated link recovery (i.e. information retrieval, traceability-mining rules, etc.). It features the sub-criteria of *Approach* and *Degree of Automation*.

**(C5) Link-Maintenance:** determines whether the tool supports the maintenance of traceability relations, how this is accomplished, and to which extent it is automated. As *Link-Creation*, it provides the sub-criteria *Approach* and *Degree of Automation*.

**(C6) Link-Storage:** determines how and where traceability links are stored, i.e. whether an internal repository, a dedicated external repository or external files are used. It is a useful criterion when a stand-alone tool is preferred and no external database or repository tool shall be required. In combination with our seventh criterion (see below), it provides

| Main Application | Interoperability | Supported Artifacts | Link-Creation | Link-Maintenance | Link-Storage | Link-Access | Link-Representation | | Required Platform | General Information |
|---|---|---|---|---|---|---|---|---|---|---|
| | Supported Formats | Requirements | Approach | Approach | Internal Repository | Query | Internal | External | OS | Maturity |
| | Supported Standards | Architecture | Degree of Automation | Degree of Automation | External Repository | Browsing | Link-Types | Graph | Tools | License |
| | Supported Tools | Source Code | | | External Files | | Link-Metadata | Matrix | | Type |
| | Multi-user Support | Test Cases | | | | | Multiplicity | Hyperlinks | | |
| | | Others | | | | | | Text | | |

Table 1: The classification criteria of our framework.

useful information if and how easily other tools can access (internal) traceability information.

**(C7) Link-Access:** determines how traceability links can be accessed by users of the tool, i.e. whether a traceability browser or a query-mechanism is used. The availability of a query-mechanism is especially important when links are stored in a repository or database and need to be accessed according to certain query-criteria.

**(C8) Link-Representation:** is used to classify how traceability links are presented to the user and how they are defined internally. Thus, it is refined as follows.

*Internal Link-Representation* classifies the internal structure of traceability links and is further refined into *Link-Types* (Does the tool support typed links?), *Link-Metadata* (Which metadata can be attached to links?), and *Multiplicity* (Does the tool support binary or n-ary links?).

*External Link-Representation* determines how traceability links are represented to the user of the tool, i.e. whether a graph-based approach, a traceability matrix, embedded hyperlinks or other textual representations are used [15].

**(C9) Platform Requirements:** provides details about the platform environment required by the tool to operate. It distinguishes between *required operating system*, and *required tools*. *Platform requirements* is a good candidate to be used as first exclusion criterion when searching for a suitable tool.

**(C10) General Information:** contains useful information which describe the general nature of a tool. It is further refined into *Maturity* (e.g. *Prototype*), *License*, and *Type* (stand-alone tool, Plug-in). This criterion might be important if only mature tools shall be considered and stand-alone tools are preferred.

Our classification framework and all of its criteria are further summarized by Table 1. A mapping of our criteria to the requirements defined in Section 2.4 is presented by Table 2.

| | | Requirements | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *R1* | *R2* | *R3* | *R4* | *R5* | *R6* | *R7* | *R8* | *R9* | *R10* |
| Criteria | *C1* | X | | X | | | | | | | |
| | *C2* | | X | | | X | | X | X | | |
| | *C3* | | | | | X | | | X | | |
| | *C4* | X | | | | | | | | | X |
| | *C5* | X | | | | | | | | | X |
| | *C6* | | | | X | | X | | | | |
| | *C7* | | | | X | | | | | | |
| | *C8* | | | X | X | | X | | | | |
| | *C9* | | X | | | | | | | X | |
| | *C10* | | X | | | | | | | | |

Table 2: A mapping of our criteria to our unified requirements.

## 4 Applications in Practice

In this section we demonstrate the applicability of our framework. Thereby we incorporate the guidelines for practitioners to make decisions about traceability tools as outlined in [7]. Following these guidelines, the stakeholder specific problem domain must first be understood and specified, so that the most relevant criteria can be identified. We use our criteria for classifying traceability tools and further demonstrate how our framework can be used to find and compare suitable tools.

### 4.1 Identifying the Problem Domain

The main application of our framework is to enable different stakeholders to find a tool which suits their needs and requirements (see Section 2.3). We exemplary describe three stakeholder specific scenarios and their respective problem domains: a maintenance programmer implementing a change request, a researcher who requires an oracle to compare the results of her traceability mining tool, and a project manager assessing the state of a project.

**Scenario 1:** a *maintenance programmer* is commissioned with implementing a change request. She faces the following challenges: identifying the parts of Java code which require rework, identifying the impacts of the change, updating all related UML diagrams, and rerunning all JUnit tests covering the changed artifacts. Furthermore, she is working with a Linux operating system and is using the Eclipse IDE. She does not want to switch to another operating system and prefers a tool which can be integrated into Eclipse. Additionally, she prefers traceability links to be visualized by a matrix. She would then consider the following criteria as important: *Main*

*Application*, *Supported Artifacts*, *External Link-Representation/Visualization*, *Platform Requirements [Required Tools/Frameworks, Operating System]*, *General Information [Type]*.

**Scenario 2:** a *researcher* developed a new traceability mining tool and wants to assess its performance. She requires an oracle to compare the output of her tool. The oracle should be an open source tool, and support all tools she used to model her case study using UML and URN (e.g. Eclipse UML2 Tools[7], jUCMNav[8]). She would use the following criteria: *General Information [License]*, *Link Creation [Automated]*, *Supported Artifacts*, *Interoperability [Supported Tools]*.

**Scenario 3:** a *project manager* is in charge of a medium sized software development project. Her main focus is to keep an eye on project progress, project costs, and quality of the final software product. Main purpose of monitoring these aspects is to continuously evaluate the current project status and, if necessary, to intervene regulatory. Another purpose is to collect, aggregate, and properly visualize tailored information to inform project stakeholders about certain aspects of the project status in an efficient way. She creates and maintains project plans (time, resource planning) with MS Project, which is already integrated into MS SharePoint document libraries and workflows. Therefore she prefers a tool that can be integrated into SharePoint/Project as well. She further requires sophisticated traceability querying and data preparation mechanisms in order to provide meaningful reports, tailored to the specific needs of readers. Thus she would use these criteria: *Main Application*, *Supported Artifacts*, *Interoperability [Supported Tools]*, *Link-Storage*, *Link-Access*, *External Link-Representation/Visualization*.

## 4.2 Classifying Tools

In order to demonstrate the applicability of our framework regarding the classification of tools, we choose two research tools and one commercial tool, and classify them according to our criteria. The selection is based on the availability of information (i.e. trial versions, documentation, publications) and the main application. The complete classification of all three tools is listed in Table 3.

*EMFTrace*[9] is our own research prototype tool, developed for traceability mining among different types of artifacts. The main application is to assist developers with impact analysis activities. A rule-based mechanism for traceability detection is utilized.

*Yakindu Requirements* is a requirements traceability tool which is embedded into the Yakindu tool environment[10] for Eclipse. It provides a set of trace-point adapters to support end-users with creating and maintaining requirements traceability links.

*traceMaintainer*[11] is a prototype tool developed to automatically detect software refactoring activities and thereby maintain existing traceability links.

Please note that our framework is still subject of ongoing research and improvement. We are currently conducting a field study to elicitate the traceability practices of several software developers from different domains. Our goal is to continuously evaluate and refine our criteria under the aspects of practical relevance and applicability. We further want to set up and expand a traceability tool database based on our classification framework. The output of our field study as well as community input is intended to form the content of the database.

| | | | Tools | | |
|---|---|---|---|---|---|
| | | | EMFTrace | Yakindu Requirements | traceMaintainer |
| Criteria | Main Application | | Impact Analysis | Requirements Traceability | |
| | Link-Storage | Internal Repository | | | |
| | | External Repository | Yes | Yes | Yes |
| | | External Files | | Yes | |
| | Internal Link-Representation | Link-Types | 77 | Yes | Yes |
| | | Link-Metadata | Time stamps, Decisions | | |
| | | Link-Multiplicity | Binary | Binary | Binary |
| | External Link-Representation / Visualization | Graph | Yes | Yes | Yes |
| | | Matrix | | | |
| | | Hyperlinks | Yes | | |
| | | Textual Representation | | Yes | Yes |
| | Link-Access | Query | | Yes * | |
| | | Browsing | Yes | Yes | |
| | Link-Creation | Approach | Rule-based | Manual | |
| | | Degree of Automation | Fully | None (but can be extended by API *) | |
| | Link-Maintenance | Approach | | Manual with decision support | Rule-based, Event-driven |
| | | Degree of Automation | | None (but can be extended by API *) | Fully |
| | Interoperability | Supported Formats | XMI, XML | XMI, XML | XML |
| | | Supported Standards | ITU-T Z.151 | ReqIF 1.0 | |
| | | Supported Tools | Visual Paradigm, jUCMNav, UML2 Tools, Protégé, pure::variants | ProR, Papyrus UML, Artop *, Yakindu SCT *, Yakindu Damos * | Sparx Enterprise Architect, ARTiSAN Studio, EXTESSY ToolNet |
| | | Multi-User Support | Yes | | |
| | Supported Artifacts | Requirements | URN, UML | ReqIF, UML | SysML, UML |
| | | Architecture | UML | UML | UML |
| | | Source Code | Java | Java, C++ | |
| | | Tests | JUnit | | |
| | | Other | Feature Models, BPMN | AUTOSAR * | |
| | General Information | Maturity | Prototype | Beta-Release | Prototype |
| | | License | EPL | EPL / Commercial | Experimental |
| | | Type | Plug-in | Plug-in | Stand-alone |
| | Platform Requirements | Operating System | Any OS supported by Eclipse | Any OS supported by Eclipse | Any OS supported by .Net/Mono |
| | | Required Tools / Frameworks | Eclipse, EMFStore | Eclipse, CDO * | .Net/Mono |

Table 3: *EMFTrace*, *Yakindu Requirements* (* available in commercial version only), and *traceMaintainer* classified according to our framework.

## 4.3 Searching Tools

The following section illustrates exemplary how our classification framework could be used by a software developer. A *maintenance programmer*, as described

[7] http://www.eclipse.org/modeling/mdt/downloads/
[8] http://www.ohloh.net/p/11712
[9] https://sourceforge.net/projects/emftrace/
[10] http://www.itemis.de/yakindu

in Scenario 1, considers the following criteria as important: *Main Application*, *Supported Artifacts*, *External Link-Representation/Visualization*, *Platform Requirements [Required Tools/Frameworks, Operating System]*, *General Information [Type]*. To address and resolve her concrete problem she would therefore use a query as described in Table 4 to search for relevant tools within the database.

| **Criterion** | **Value** |
|---|---|
| Main Application | Impact Analysis, Feature Localization, Regression Test Selection |
| Supported Artifacts | Java, UML, JUnit |
| External Link Representation/Visualization | Matrix |
| Platform Requirements [Operating System] | Linux |
| Platform Requirements [Required Tools/Framework] | Eclipse |
| General Information [Type] | Plug-in |

Table 4: Possible query for Scenario 1

## 5 Conclusion and Future Work

In this paper, we propose a framework for classifying, searching, and comparing traceability tools. Our framework is comprised of ten criteria, which are based on an analysis of related work on comparing three different types of tools, structured CASE tool characteristics as standardized by ISO/IEC, and traceability benchmarks. The proposed criteria were further refined and extended after studying five typical stakeholders of traceability tools according to their needs and roles during software development.

We demonstrated the applicability of our framework in two steps. First, we have shown how the classification of traceability tools can be accomplished by classifying three tools according to our criteria. Second, we illustrated how our criteria can be used to search for a traceability tool for a certain task, given that a list of tools which are classified according to our framework is available. Future work will take two directions. We need to further validate the applicability of our framework by classifying more traceability tools according to its criteria. Therefore we conduct a field study on whether and how traceability is used in the industrial context, how traceability is incorporated in existing software development tools and processes, and what problems still exists with traceability. Our main goals however are to build a database of traceability tools, classified according to our criteria, to enable other researchers and practitioners to search and compare traceability tools and to continuously refine our criteria framework.

## References


[1] A. Bianchi, A. R. Fasolino, and G. Visaggio. An exploratory case study of the maintenance effectiveness of traceability models. In *Proc. of 8th Intl. Workshop on Program Comprehension*, pages 149 – 158, 2000.

[2] E. B. Charrada, D. Caspar, C. Jeanneret, and M. Glinz. Towards a benchmark for traceability. In *Proceedings of the 12th Intl. Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution*, pages 21–30, 2011.

[3] J. Cleland-Huang, A. Czauderna, A. Dekhtyar, O. Gotel, J. Huffman Hayes, E. Keenan, G. Leach, J. I. Maletic, D. Poshyvanyk, Y. Shin, A. Zisman, G. Antoniol, B. Berenbach, A. Egyed, and P. Maeder. Grand challenges, benchmarks, and tracelab: Developing infrastructure for the software traceability research community. In *Proc. of the 6th Intl. Workshop on Traceability in Emerging Forms of SE*, 2011.

[4] C. de Gea, M. Juan, J. Nicolás, J. Alemán, A. Toval, C. Ebert, and A. Vizcaíno. Requirements engineering tools. *Software, IEEE*, 28(4):86–91, 2011.

[5] A. Dekhtyar, J. H. Hayes, and G. Antoniol. Benchmarks for traceability? In *Proceedings of Traceability in Emerging Forms of Software Engineering*, 2007.

[6] A. Egyed, S. Biffl, M. Heindl, and P. Grünbacher. A value-based approach for understanding cost-benefit trade-offs during automated software traceability. In *Proc. of the 3rd Intl. workshop on Traceability in emerging forms of SE*, pages 2–7. 2005.

[7] O. Gotel and P. Mäder. Acquiring tool support for traceability. *Software and Systems Traceability*, pages 43–68, 2012.

[8] P. Letelier. A framework for requirements traceability in UML-based projects. In *Proc. 1st Intl. Workshop on Traceability in Emerging Forms of Software Engineering*, pages 32–41, 2002.

[9] M. Lindvall and K. Sandahl. Practical implications of traceability. *Softw. Pract. Exper.*, 26:1161–1180, October 1996.

[10] M. Lindvall and K. Sandahl. Traceability aspects of impact analysis in object-oriented systems. *Journal of Software Maintenance: Research and Practice*, 10:37–57, January 1998.

[11] P. Mäder, O. Gotel, and I. Philippow. Rule-based maintenance of post-requirements traceability relations. In *Proceedings of the 2008 16th IEEE International Requirements Engineering Conference (RE '08)*, pages 23–32, Washington, DC, USA, 2008. IEEE Computer Society.

[12] J. I. Maletic, E. V. Munson, A. Marcus, and T. N. Nguyen. Using a hypertext model for traceability link conformance analysis. In *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering*, pages 47–54, 2003.

[13] T. Olsson and J. Grundy. Supporting traceability and inconsistency management between software artifacts. In *Proc. of the Intl. Conference on Software Engineering and Applications*, pages 63–78, 2002.

[14] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M. Ali Babar. A comparative study of architecture knowledge management tools. *Journal of Systems and Software*, 83(3):352–370, 2010.

[15] S. Winkler and J. von Pilgrim. A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling*, 9(4):529–565, 2010.