Towards Practical Software Traceability

Hazeline U. Asuncion
Institute for Software Research
University of California, Irvine
Irvine, California 92697, USA
+1-949-824-2703
http://www.isr.uci.edu/~hasuncio/
hasuncion@ics.uci.edu

ABSTRACT
The importance of software traceability to software development is recognized by researchers and practitioners; yet, current approaches fall short of providing effective traceability in practice. An analysis of reported difficulties with traceability reveals that interacting factors from the economic, technical, and social perspectives hinder traceability. Motivated by the multi-faceted traceability problem, we combine architecture-centric stakeholder-driven traceability with open hypermedia, and we use insights from e-Science to guide our approach. We highlight expected contributions and discuss evaluation plans. Finally, we distinguish our approach from related research and technologies.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management – *Life cycle;* D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement – *Documentation.*

General Terms
Management, Documentation

Dissertation Advisor
Richard N. Taylor – taylor@ics.uci.edu

1. SOFTWARE TRACEABILITY
Illuminating the various interrelationships of artifacts within a project, software traceability has been recognized by both researchers and practitioners as a key factor for improving software development [8, 9]. Potential benefits to traceability include better impact analysis, lower maintenance costs, and better assessment of product quality [13, 17]. In some cases, traceability is needed to comply with internal standards and external regulations [4, 17].

Despite these advantages, traceability is difficult to achieve in practice. Current approaches are generally infeasible [4]. Most of the manual approaches require high overhead and are viewed by software engineers as imposed work [13]. Meanwhile, automated techniques provide limited semantics and thus do not deliver all the expected benefits [19]. To better understand why these approaches fall short, we re-examine the software traceability problem.

ICSE'08, May 10–18, 2008, Leipzig, Germany.
ACM 978-1-60558-079-1/08/05.

1.1 The Multi-faceted Traceability Problem
A survey of reported difficulties with traceability [5] reveals that many interacting factors hinder effective traceability in practice. These factors include high costs [13], complex artifact interrelationships [2], heterogeneity of artifacts and tools [8], and varied stakeholder interests [17]. Interestingly, these factors, reflecting the economic, technical, and social perspectives, are consistent with difficulties encountered in previous work [4].

1.2 Insights from e-Science
To find fresh approaches to achieving traceability, we glean insights from e-Science [5]. A domain with similarities to software engineering, e-Science is characterized by distributed global collaborations among scientists using large-scale data sets and heterogeneous computational resources [18]. Data provenance techniques in e-Science enable tracing data products across an entire experiment lifecycle. A majority of the provenance systems we surveyed are successfully used in practice. We present a few of the insights that influence our approach. (While the software traceability problem is not bounded by physical laws as in e-Science, we use insights that are generally applicable to software development.)

Provenance collection is stakeholder-driven. In data provenance systems, users control provenance collection. Users determine the information to capture, the frequency of capture, and the level of granularity. Users may also maintain their own perspective on data provenance owned by other groups to produce custom provenance views.

Provenance systems enable local ownership and global access. In some systems, local groups have complete control over their provenance information. At the same time, provenance information is visible to external groups.

Reasoners help in automatically inferring relationships. A capability afforded by some provenance systems is the ability to reason, i.e. analyze, query, and browse captured

provenance. Specialized reasoners like inference engines can infer semantic relationships between objects [6].

1.3 Research Hypotheses and Approach

Given the problem analysis and insights from e-Science, we hypothesize the following:

Software traceability across the entire software lifecycle can be achieved if economic, technical, and social factors are considered simultaneously. Difficulties in one perspective may be affected by difficulties from other perspectives. We believe that a practical technical approach must consider the economic and social perspectives.

Architecture-centric traceability, i.e. using the architecture as the primary artifact to trace all other artifacts, effectively captures the key concepts to trace. Encapsulating the resolution of stakeholders' principal concerns, the architecture provides an effective centerpiece for tracing concepts of interest to all stakeholders. We posit that the architecture is the *right level of discourse* in tracing concepts in a software product. Requirements are abstract and often volatile while code is too low level to serve as a primary organizing concept. We also posit that the architecture is the *essential level of discourse* since it is product-based.

Concepts from open hypermedia, such as n-ary first class links and lightweight hypermedia tool adapters, can be applied to facilitate the efficient capture and management of relationships between artifacts. Hypermedia links with lightweight tool adapters cross heterogeneous tools at reasonable development costs. Stored outside the artifacts they connect, n-ary first class links enable tracing heterogeneous artifacts that are maintained in diverse formats with different tools. Lightweight hypermedia tool adapters can encapsulate existing trace recovery techniques and leverage the native hypertext support within some tools to capture and manage trace links.

Stakeholder-driven traceability can be supported by enabling stakeholders to define and maintain their trace relationships using external hypermedia links and custom trace views. Stakeholders determine the artifacts to trace to the architecture, the level of granularity, and the types of trace relationships to capture. Stakeholder-driven traceability also empowers users to directly benefit from their traced artifacts with custom trace views. Since this approach closes the gap between the capture and actual usage of trace links, links can be updated as they are used, minimizing link deterioration.

2. SOLUTION AND CONTRIBUTIONS

We expect that our approach of architecture-centric stakeholder-driven traceability with open hypermedia will provide the following contributions: efficient capture of semantically-rich trace links, increased accessibility of artifacts across different groups and tools, support for varied stakeholder interests, and ease of trace link maintenance. We will implement our approach within ArchStudio [12].

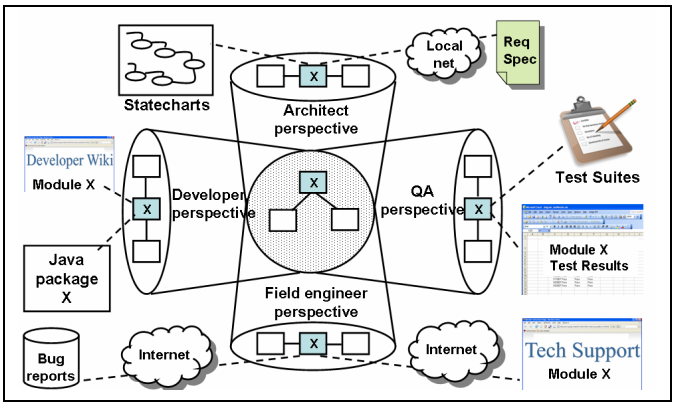

Figure 1: Architecture model as the hub of artifact tracing. Stakeholder-defined trace links support stakeholders' tasks.

Completely integrated into the Eclipse development environment, ArchStudio is a mature product-quality environment for architecture-centric development.

2.1 Efficiently Capture Trace Links

Motivated by the economic perspective, the efficient capture of trace links entails modeling artifacts and artifact relationships as well as integrating in situ link capture techniques with recovery techniques. We now elaborate these approaches.

An artifact relationship is represented as a trace link while different views of artifacts are represented as link endpoints. Since we propose to use the architecture model as the hub of tracing artifacts (see Figure 1), first class trace links will be stored in xADL [11], a modularly extensible, XML-based architecture description language. Trace links may be customized using xADL extensions.

To minimize the overhead in capturing trace links, we propose to capture trace links as a side-effect to stakeholders' development tasks. Stakeholder-defined rules (see Figure 2) along with GUI capture tools can be used to capture links while artifacts are generated or modified. Rules can also be used to automatically add semantic information to trace links (e.g. relationship type). We plan to combine these trace capture techniques with trace recovery techniques (e.g. code search [14], software repository mining [21], and information retrieval (IR) techniques [15]) and to integrate these techniques with open hypermedia tool adapters.

2.2 Increase the Accessibility of Artifacts

Motivated by the social and technical difficulties of the separation of artifacts across different groups and different tools, we increase accessibility in the following ways.

To increase accessibility across different groups, we plan to explore mechanisms for sharing trace links (e.g. link import/export, link synchronization) as well as managing links from multiple sources (e.g. aggregating trace links, removing duplicate links, etc.).

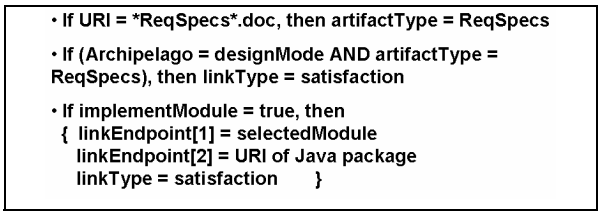

Figure 2: Sample Artifact/Link Classifier Rules

To increase accessibility across different tools, we plan to provide unified user access to the architectural model and related artifacts through Archipelago, a graphical editor in ArchStudio. We also plan to use hypermedia adapters which may either leverage the native hypertext capabilities already present in some tools (e.g. Adobe Acrobat, MS Word) or may be custom built through plug-ins, macro programs, or other extension mechanisms (e.g. Eclipse).

2.3 Support Varied Stakeholder Interests

Motivated by the social perspective, we plan to support the varied traceability interests of stakeholders by allowing stakeholders to maintain their external links in a xADL file. Stakeholders may define custom rules to capture and classify links (see Figure 2). We also plan to add a mashup [16] visualization tool to ArchStudio and leverage third party XML tools (e.g. Cocoon [3]). Thus, stakeholders can dynamically create custom views of the structural model through the aggregation of the traced information.

2.4 Easily Maintain Trace Links

Motivated by the economic perspective, we enable easier link maintenance in several ways. Note that trace links anchored on the architecture model are more stable and less likely to deteriorate than trace links centered on requirements. When the architectural model does change, stakeholders can automatically carry over their custom links into the new architectural model for the preserved modules. Along with the architecture model, trace links may be versioned in a configuration management system. Distributed link management by stakeholders as well as automated link management by open hypermedia adapters also lower the overhead for link maintenance.

3. EVALUATION

We will evaluate our approach along three key aspects: feasibility, incurred overhead, and perceived utility of information captured. We will evaluate these aspects in ArchStudio development, informal case studies in undergraduate project courses, and industry use, with each domain posing a different set of challenges.

3.1 ArchStudio

We will apply the approach to our own development environment, ArchStudio, used extensively at UCI, in several companies, and at dozens of universities. We will try to capture and to use the relationships inherent in the extensive body of information that accompanies the system to assist the on-going development of ArchStudio. We will primarily conduct a technical evaluation of the following capabilities: the ability to trace heterogeneous artifacts in different editors, the ability to adequately model different artifact relationship types, and the ability to trace artifacts at different levels of granularity and abstraction. We also plan to release a Beta version to the community of existing users. We will conduct online surveys to determine the capabilities of traceability systems users are able to build with our approach.

3.2 Informal Case Studies in Project Courses

We will evaluate our approach in the context of undergraduate project courses. The challenge in this context is the level of user experience in software development, and the short time frame (ten weeks) in which teams of four or five students implement a sizable software system. In this context, we will assess the overhead in establishing and maintaining trace links, the precision and recall rates of captured trace links, the accuracy of automatically recorded link semantics, and the performance overhead of ArchStudio in the automatic capture of trace links. We will examine student project logs and conduct informal student interviews.

3.3 Industry Use

We will conduct field trials of our tool with our current industry ties. We already built a successful traceability tool for Wonderware, an industrial automation company and a business unit of Invensys Systems, Inc. [4], and ArchStudio is in daily use within Boeing Aerospace. The unique challenges posed in these contexts are accommodating company-specific practices in developing software and tracing numerous artifacts. We will trace artifacts owned by different groups and establish links that support their specific development tasks. We will use metrics along the economic, technical, and social perspectives. We will conduct surveys and interviews to solicit answers to the following questions: Is our approach cost-prohibitive (e.g. usual tasks have been neglected due to time spent in traceability tasks)? Does it support stakeholders in their development tasks? Is the traced information useful? Is it difficult to adapt our tool to existing company tools? Is it easy to trace artifacts across different tools and different groups? We will compare the gathered responses with existing industry case studies such as [17]. The success can be gauged by the company's continued usage after our evaluation period has ended.

4. RELATED WORK

We distinguish our work from design rationale capture, retrieval techniques, and current link technologies. Design rationale capture assists designers by supporting reflection, communication among stakeholders, and analysis of past decisions [10]. While our approach supports design rationale capture by facilitating access to rationale, we do not focus on reflection or analysis. A related approach, Architecture

Rationalization Method (ARM) [20], captures rationale and establishes links from requirements to design. Catering mainly to architects, ARM assumes the existence of requirements prior to design. Meanwhile, our approach does not assume the presence of requirements nor does it favor a development process over another. Our approach caters to a wide range of users and enables traceability across a wider range of artifacts. Other techniques like software repository mining [21] or IR technology [15] support specialized searches on software artifacts. Hipikat uses several heuristics to automatically identify links between artifacts [7]. We can integrate these techniques with other trace capture and recovery techniques. Finally, while XML Topic Maps [1] enable manual capture of links, our approach of using hypermedia tool adapters allows for the automatic capture and management of links.

5. PROGRESS OF RESEARCH

We have completed the problem analysis and have surveyed data provenance systems from e-Science. We have also built a successful software traceability tool at a leading industrial automation software development company [4] which is geared to support a waterfall model with short lifecycles. We are currently building traceability support into ArchStudio.

6. CONCLUSIONS

Practical traceability solutions remain elusive because interacting factors such as high costs, complex artifact interrelationships, heterogeneous tools, and varied stakeholder interests are not adequately addressed. By *centering traceability links on the architecture*, traced concepts carry significance to all stakeholders and traceability relationships are grounded in the product. By *supporting stakeholder-driven traceability*, stakeholders directly benefit from traced information. By *using n-ary first class links to represent trace relationships*, semantically-rich links can be expressed over heterogeneous artifacts residing in different authority domains on the Internet. Our approach takes into account the multi-faceted traceability problem and moves us closer to achieving software traceability in practice.

7. ACKNOWLEDGEMENTS

Many thanks to my advisor for encouraging me to study data provenance in e-Science and for his keen insight into various technologies. Thanks to the anonymous reviewers for their feedback. Effort partially funded by the National Science Foundation under grants IIS 0205724 and CNS-0438996.

8. REFERENCES

[1] Topic Maps. <http://www.topicmap.com/>.

[2] Anderson, K.M., Sherba, S.A., et al. Towards Large-Scale Information Integration. In Proc. of the 24th ICSE. Orlando, Florida, May, 2002.

[3] Apache Software Foundation. Cocoon. <http://cocoon.apache.org>.

[4] Asuncion, H., François, F., et al. An End-To-End Industrial Software Traceability Tool. In Proc. of the 6th Joint Meeting of the ESEC/FSE. Dubrovnik, Croatia, Sep, 2007.

[5] Asuncion, H. and Taylor, R.N. Establishing the Connection Between Software Traceability and Data Provenance. ISR, UC, Irvine, Tech Report UCI-ISR-07-9, Nov, 2007.

[6] Cheung, K. and Hunter, J. Provenance Explorer - Customized Provenance Views Using Semantic Inferencing. In Proc. of the 5th Int'l Semantic Web Conference. Athens, GA, Nov 5-9, 2006.

[7] Cubranic, D., Murphy, G.C., et al. Hipikat: a Project Memory for Software Development. *IEEE TSE*. 31(6), p. 446-65, 2005.

[8] Domges, R. and Pohl, K. Adapting Traceability Environments to Project Specific Needs. *CACM*. 41(12), p. 54-62, 1998.

[9] Evans, M. SPMN Director Identifies 16 Critical Software Practices. *CrossTalk, The Journal of Defense Software Engineering*. Mar, 2001.

[10] Horner, J. and Atwood, M.E. Design Rationale: the Rationale and the Barriers. In 4th Nordic Conf on HCI: Changing Roles: Oslo, Norway, 2006.

[11] Institute for Software Research. xADL 2.0. <http://www.isr.uci.edu/projects/xarchuci/>, UC, Irvine, 2005.

[12] Institute for Software Research. ArchStudio 4. <http://www.isr.uci.edu/projects/archstudio/>, UC, Irvine, 2006.

[13] Jarke, M. Requirements Tracing. *CACM*. 41(12), p32-36, Dec 1998.

[14] Lopes, C.V. and Baldi, P. The Sourcerer Project. <http://sourcerer.ics.uci.edu/>.

[15] Marcus, A. and Maletic, J.I. Recovering Documentation-To-Source-Code Traceability Links Using Latent Semantic Indexing. In Proc. of 25th ICSE. Portland, OR, May 3-10, 2003.

[16] Merrill, D. Mashups: The New Breed of Web App. <http://www.ibm.com/developerworks/xml/library/x-mashups.html>, 2006.

[17] Ramesh, B., Powers, T., et al. Implementing Requirements Traceability: A Case Study. In Proc. of the 1995 Intl. Symp on Req Eng (RE'95). p. 89-95, York, UK, Mar 27-29 1995.

[18] Senior, I. e-Science Definitions. <http://e-science.ox.ac.uk/public/general/definitions.xml>, 2002.

[19] Spanoudakis, G. and Zisman, A. Software Traceability: A Roadmap Advances in Software Eng & Knowledge Eng. Chang, S.K. ed. 3, World Scientific Publishing, 2005.

[20] Tang, A. and Han, J. Architecture Rationalization: A Methodology for Architecture Verifiability, Traceability and Completeness. Proc. of 12th IEEE Int'l Conf & Workshop on the Eng of Comp-Based Systems. Greenbelt, MD, Apr 4-7, 2005.

[21] Zimmermann, T., Weibgerber, P., et al. Mining Version Histories to Guide Software Changes. In Proc. of the 26th ICSE Edinburgh, UK, May 23-28 2004.