IR-based Traceability Recovery as a Plugin – An Industrial Case Study

Markus Borg
Department of Computer Science
Lund University
Lund, Sweden
markus.borg@cs.lth.se

Large-scale software development is a complex undertaking and generates an ever-increasing amount of information. To be able to work efficiently under such circumstances, navigation in all available data needs support. Maintaining traceability links between software artefacts is one approach to structure the information space and support this challenge. Several researchers have proposed traceability recovery by applying IR methods, based on textual similarities between artefacts. Early studies have shown promising results, but no large-scale in vivo evaluations have been made. Currently, there is a trend among our industrial partners to collect artefacts in a specific new software engineering tool. Our goal is to develop an IR-based traceability recovery plugin to this tool. From this position, in the environment of possible future users, the usefulness of supported findability in a software engineering context could be explored with an industrial validity.

software engineering, traceability, information retrieval, findability, IR evaluation

1. INTRODUCTION

In large-scale software development, coordination between different organizational units is a key success factor to develop high-quality products on time and within budget. Software development results in a myriad of information entities. Apart from the source code itself, requirements and design specifications at variousabstraction levels, test descriptions, test results and defect reports are examples of produced software artefacts. The term software artefact refers to any piece of information, a final or intermediate work product, which is produced and maintained during software development (Kruchten, 2004). Software artefacts are the tangible results of the development process. They are typically of volatile nature and subject to version control.

Developing techniques to navigate all this growing information is crucial. Current state-of-practice is to structure the information space by manually maintaining traceability links between software artefacts. This is widely recognized as an important factor for efficient development, since it supports verification, change impact analysis, program comprehension and software reuse (Antoniol et al., 1999). Lack of traceability has been identified as one of the top factors causing delays in software engineering projects (Dömges and Pohl, 1998).Since a traceability link can be established between any software artefacts, defining a suitable trace granularity is an important decision in a development project (Cleland-Huang et al., 2007).

Software artefacts can consist of source code, UML models, diagrams, state machines, graphics, binaryfiles etc. However, text in natural language is the common form of information representation during all development phases (Marcus and Maletic, 2003). Also source code contains natural language content in identifiers and comments. Consequently several researchers have proposed standard IR approaches to semi-automatically trace software artefacts by presenting candidate links. The trend in this research has been to hunt recall and precision values on a rather limited set of small publicly available datasets, often from student projects or the open source domain (Huffman Hayes et al., 2006, Zou et al., 2006, Capobianco et al., 2009). Recently, case studies have been conducted using proprietary data from the industry, but they are still in minority.

The goal of our research is not primarily to study how IR methods can be improved and configured to perform better in an industrial setting, but rather to evaluate the IR-based approach in general and study how software engineers can benefit from increased findability through traceability

recovery.To reach that goal, we plan to implement the functionality as a plugin in an existing tool.

2. RELATED WORK

The most cited definition of traceability has been given by Gotel and Finkelstein (1994):

> "Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forward and backward direction (i.e. from its origin, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases)"

Two main areas where traceability is beneficial are (1) *process compliance and product improvement* and (2) *software understanding and reuse* (Spanoudakis and Zisman, 2005). The former is related to recommendations or requirements on the development process and include standards such as IEEE Std. 830-1998 and laws governing safety-critical systems such as US Sarbones-Oxley Act of 2002 (Cleland-Huang et al., 2007). Traceability in the context of (2) software understanding and reuse supports maintenance and reengineering of legacy systems (Antoniol et al., 2002).

Fiutem and Antoniol (1998) did early work on recovering traceability links between design and source code. They used basic string comparisons and edit distances to suggest links between design documentation and source code. In the following years, Antoniol et al. (2002) continued by using the Vector Space Model (VSM) and probabilistic models to recover traceability links between source code and textual documentation in natural language. Marcus and Maletic (2003) introduced Latent Semantic Indexing (LSI) to recover traceability, also between code and documentation. They also showed that LSI can achieve good results without the need for stemming.

Huffman Hayes et al. have applied both VSM and LSI in traceability recovery and have also studied less technical aspects such as how software engineers participate in the tracing loop (Huffman Hayes and Dekhtyar, 2005). Also DeLucia et al. (2009) have assessed the usefulness of supporting traceability recovery in software engineering, in a controlled experiment with students.

The risk of spending too much effort on improving techniques for document retrieval without considering the actual needs of the users has been knownfor decades (Lancaster,1968). Directing effort on increasing the size of datasets instead of spending time on optimizing algorithms on small corpora is important, since methods might converge (Banko and Brill,2001). Recently, Oliveto et al. (2010) presented a case study on traceability recovery where VSM, LSI and the Jensen-Shannon method were compared and the results were almost equivalent.

3. EXPLORING STATE-OF-PRACTICE AND STATE-OF-ART

A large in-depth exploratory interview study was initiated in 2009 to investigate software engineers' views on alignment between requirements and test activities. We have conducted 30 interviews in 6 different companies with interviewees representing different roles in the development process. The overall goal of the study was to better understand the context to focus our future research. Our study identified poor tool support, information distributed in separate systems with poor interoperability and lack of traceability as contributing factors of misalignment (Sabaliauskaite et al.,2010).

To investigate the state-of-art of IR-based traceability recovery, we are working on a systematic mapping study (Kitchenham, 2004). Preliminary results from the meta-analysisshow a need for in vivo evaluations of the approach; most previous evaluations involving human subjects have been conducted in university settings with student subjects. The final step of the study will map empirical results according to IR techniques, validity of datasets and types of traceability links established.

Another parallel activity, a master thesis project, found the public availability of the research prototypes to be low. The thesis evaluated IR-based tools for traceability recovery using requirements and test case descriptions collected from safety critical development in the domain of power and automation (Brodén, 2011).

4. DEVELOP PLUGIN IN STATE-OF-PRACTICE TOOL

Some of our industrial partners are working on introducing *HP Quality Center* (QC) as a new software engineering tool. A direct outcome of this transition will be that requirements, test cases and defect reports will be accessible in the same tool. This means the issue of poor tool interoperability highlighted by practitioners in our case study will no longer be a major obstacle. Another major advantage of this tool change in industry is that QC has good support for plugin development, thus it can be used as a test bed for our approach. This would enable us to implement an IR-based traceability tool within the system, right in the centre of the information hub.

5. EVALUATE APPROACH IN INDUSTRIAL CASE STUDY

The aim of this study will be to evaluate how well the IR-based approach to traceability recovery works in a real industrial setting. With the plugin in place, we will be able to study the performance of IR-based approaches for traceability recovery with an industrial validity. It will also enable us to study software engineers and their artefacts without introducing any additional external tools. The focus will be less on recall and precision, since the real question is to what extent the approach actually supports engineers. Instead aspects such as how much you benefit from improved findability of traceability information, how it affects the way engineers work, how much time can be saved etc. should be addressed.

A suitable method for the empirical evaluation is a case study (Runeson and Höst,2009). In vivo studies are hard to conduct as experiments, since the level of control usually is too low. Collected data will include tool usage statistics complemented by answers from interviews and a questionnaire distributed among involved practitioners. The plugin solution would also simplify expanding the study to multiple companies.

ACKNOWLEDGEMENTS

This work was funded by the Industrial Excellence CenterEASE - Embedded Applications Software Engineering[1].

6. REFERENCES

Antoniol G., Canfora G., De Lucia A. and Merlo E. (1999) Recovering code to documentation links in OO systems. 6th *Working Conference on Reverse Engineering*.

Antoniol G., Canfora G., De Lucia A., Casazza G.and Merlo E. (2002) Recovering traceability links between code and documentation.*IEEE Transaction on Software Engineering*. 28(10), 970-983.

Banko M. and Brill E. (2001) Scaling to very very large corpora for natural language disambiguation.39th *Annual Meeting on Association for Computational Linguistics.*

Brodén L. (2011) Requirements Traceability Recovery – A Study of Available Tools. *Master's Thesis,*Dept. Computer Science, Lund University, http://sam.cs.lth.se/ExjobGetFile?id=377

Capobianco G., De Lucia A., Oliveto R., Panichella A. and Panichella S. (2009) On the Role of the Nouns in IR-based Traceability Recovery. *17th International Conference on Progam Understanding*.

Cleland-Huang J., Settimi R., Romanova E., Berenbach B. and Clark S. (2007) Best Practices for Automated Traceability. *Computer*, 40(6), 27-35.

De Lucia A., Oliveto R. and Tortora G. (2009), Assessing IR based traceability recovery tools through controlled experiments, *Empirical Software Engineering*, 14(1), 57-92.

Dömges R. and Pohl K. (1998) Adapting traceability environments to project-specific needs, *Communications of the ACM*, 41(12), 52-62.

Fiutem R. and Antoniol G. (1998) Identifying design-code inconsistencies in object-oriented software: a case study. *Conference on Software Maintenance.*

Gotel O. and Finkelstein A. (1994) An Analysis of the Requirements Traceability Problem. 1st *International Conference on Requirements Engineering.*

Huffman Hayes J. and Dekhtyar A. (2005) Humans in the traceability loop: can't live with 'em, can't live without 'em. 3rd International Workshop on *Traceability in Emerging Forms in Software Engineering*.

Huffman Hayes J., Dekhtyar A. and Sundaram S.(2006) Advancing candidate link generation for requirements tracing: the study of methods. *IEEETransaction on Software Engineering*, 32(1), 4-19.

Kitchenham B. (2004) Procedures for performing systematic reviews. *Technical report*, Keele University and NICTA.

Kruchten P. (2004) The Rational Unified Process: An Introduction. Third edition. Pearson Education, Boston, MA, USA.

Lancaster F. and Climenson W. (1968) Evaluating the economic efficiency of a document retrieval system. *Journal of Documentation*, 24(1), 16-40.

Marcus A. and Maletic J. (2003) Recovering documentation-to-source-code traceability links using latent semantic indexing, 25th *International Conference on Software Engineering.*

Oliveto R., Gethers M., Poshyvanyk D. and De Lucia A. (2010) On the equivalence of information retrieval methods for automated traceability link recovery,18th *International Conference on Program Comprehension.*

Runeson P. and Höst M. (2009) Guidelines for conducting and reporting case study research in software engineering, *Empirical Software Engineering*, 14(2),131-164.

Sabaliauskaite G., Loconsole A., Engström E., Unterkalmsteiner M., Regnell B., Runeson P., Gorschek T. and Feldt R. (2010) Challenges in aligning requirements engineering and verification in a large-scale industrial context.*Requirements Engineering: Foundation for Software Quality*

Spanoudakis G. and Zisman A. (2005), Software Traceability: a Roadmap.*Handbook of Software Engineering and Knowledge Engineering*, Volume

[1] http://ease.cs.lth.se

3:Recent Advancements, World Scientific Publishing Co.
Zou X., Settimi R. and Cleland-Huang J. (2006) Phrasing in dynamic requirements trace retrieval. *30th Annual International Computer Software and Applications Conference*.