IR-based Traceability Recovery as a Plugin

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, a unique evaluation of the usefulness of supported findability in a software engineering context could be made.

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 textual information; requirements and design specifications at various abstraction levels and test descriptions, test results and defect reports to mention some. Developing techniques to navigate all this growing information is crucial.

Current state-of-practise is to structure the information space by 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, 1999). Lack of traceability has been identified as one of the top factors causing delays in software engineering projects (Domges, 1998).

Text is the common form of information representation and several researchers have proposed IR approaches to semi-automatically trace artefacts. 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. 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.

2. RELATED WORK

Fiutem and Antoniol did early work on recovering traceability links between design and source code in 1998. They used basic string comparisons and edit distances to suggest links between design and code (Fiutem, 1998). In the following years, Antoniol et al. continued by using the Vector Space Model (VSM) and probabilistic models to recover traceability links between source code and textual documentation in natural language (Antoniol, 2000). Marcus and Maletic introduced Latent Semantic Indexing (LSI) to recover traceability in 2003 (Marcus, 2003). Hayes et al. have applied both VSM and LSI in traceability recovery and have also studied less technical aspects such as how human analysts participate in the tracing loop (Hayes, 2005). Also De Lucia et al. have assessed the usefulness of supporting traceability recovery in software engineering, in a controlled experiment with student subjects (De Lucia, 2009).
The risk of spending too much effort on improving techniques for document retrieval without considering the actual needs of the users has been known for 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, 2001). Recently, Oliveto et al. presented a traceability case study where VSM, LSI and the Jensen-Shannon method were compared and the results were almost equivalent (Oliveto, 2010).

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. 30 interviews in 6 different companies have been conducted. The study identified both poor tool support, poor interoperability and lack of traceability as challenges (Sabaliauskaite, 2010). In parallel, we have been working on a systematic mapping study on IR-based traceability recovery. This study showed a need for in vivo evaluations of the approach. Another parallel activity, a master thesis project, found the public availability of the research prototypes to be low. The thesis evaluated IR-tools using documentation collected from industry.

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 developers and their information without introducing any additional external tools. The focus will be less on recall and precision, instead aspects such as how much you benefit from improved findability of traceability information, how it affects the way developers work, how much time can be saved etc. should be addressed.

A suitable method for the empirical evaluation is a case study (Runeson, 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.

4. REFERENCES

Banko M. and Brill E. (2001) Scaling to very very large corpora for natural language disambiguation. ACL '01.
Hayes J. and Dekhtyar A. (2005) Humans in the traceability loop: can't live with 'em, can't live without 'em. TEFS.
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, REFSQ.