The Benefit of Requirements Traceability When Evolving a Software Product: A Controlled Experiment

Patrick Mäder[1] and Alexander Egyed[2]

Abstract: Software traceability is a required component of many software development processes. Advocates of requirements traceability cite advantages like easier program comprehension and support for software maintenance (i.e., software change). However, despite its growing popularity, there exists no published evaluation about the usefulness of requirements traceability. It is important, if not crucial, to investigate whether the use of requirements traceability can significantly support development tasks to eventually justify its costs. We thus conducted a controlled experiment with 71 subjects re-performing real maintenance tasks on two third-party development projects: half of the tasks with and the other half without traceability. Subjects sketched their task solutions on paper to focus on the their ability to solving the problems rather than their programming skills. Our findings show that subjects with traceability performed on average 24% faster on a given task and created on average 50% more correct solutions – suggesting that traceability not only saves effort but can profoundly improve software maintenance quality.

Keywords: requirements traceability; software evolution; effect; controlled experiment; study

1 Motivation and Study

Capture and maintenance of requirements-to-code traces reflect knowledge where requirements are implemented in the code is the focus of extensive research [CHGHH$^+$14]. Despite its growing popularity [MGP09, RMK13] surprisingly little is known about its benefits. Intuitively, requirements-to-code traces should be useful for many areas of software engineering. Researchers refer to better program comprehension and support for software maintenance. Nonetheless, there exists no empirical work in which the effect of requirements traceability was measured. Does it save effort? Does it improve quality?

In this study, originally published at [ME15], we assessed whether available requirements-to-code traces improve the performance of subjects during software maintenance and evolution tasks. Therefore, we conducted an experiment involving 71 practitioners and students with a wide range of experiences. The subjects were asked to solve tasks taken from two software projects: the open source Gantt Project (47 KLOC) and the iTrust system (15 KLOC). Eight tasks were selected, covering real bug fixes and feature extensions taken from the projects' documented archives. Tasks were randomly assigned to subjects, half with and the other half without traceability. Task solutions were recorded on paper and not implemented by the subjects. We measured the performance of subjects as the time they

[1] Technische Universität Ilmenau, Software Systems Group, Ilmenau, Germany, patrick.maeder@tu-ilmenau.de

[2] Johannes Kepler University, Institute for Software Systems Engineering (ISSE), Linz, Austria, alexander.egyed@jku.at

spent to solve a task and the correctness of their solution. The selected, real maintenance tasks also provided us with a gold standard as to how the original developers solved the given tasks. Furthermore, we assessed the influence of subject experience, the kind of tasks subjects were expected to solve, and the different project domains. All subjects were not familiar with the projects – a situation commonly occurring during software maintenance and a situation under which developers are expected to benefit most from traceability.

2 Results and Conclusions

In total, subjects solved 461 tasks (i.e., 6.5 tasks per subject on average). Our findings show that subjects working on tasks with traceability performed better than subjects working without traceability. In particular, subjects with traceability performed on average 24% faster on tasks and created on average 50% more correct solutions. This demonstrates that traceability is not just a means for saving some effort but can profoundly improve the quality of the software maintenance process. There are likely many subsequent benefits such as more effective maintenance, faster time to market, or less code degradation. We also found that some tasks benefited more from traceability than others, especially with regard to the correctness of the solution. Furthermore, we found that our observations were consistent regardless of subject experience and the project domain.

The implications of this study are numerous. Traceability strongly benefits software maintenance regardless of subject experience. Though often perceived tedious and ineffective, this work demonstrates a clear, measurable performance improvement to justify traceability cost. Since this work clearly characterizes the effect of traceability, practitioners and researchers alike may use this information to better understand the cost/benefit trade-off of traceability – a point that will also be the focus of our future work.

Acknowledgments We are funded by the German Ministry of Education and Research (BMBF) grant 01IS14026B and the Austria Science Fund (FWF) grant FWF P 25289-N15.

References

[CHGHH+14] Jane Cleland-Huang, Orlena Gotel, Jane Huffman Hayes, Patrick Mäder, and Andrea Zisman. Software Traceability: Trends and Future Directions. In *Proc. 36th International Conference on Software Engineering (ICSE)*, pages 55–69, 2014.

[ME15] Patrick Mäder and Alexander Egyed. Do developers benefit from requirements traceability when evolving and maintaining a software system? *Empirical Software Engineering*, 20(2):413–441, 2015.

[MGP09] Patrick Mäder, Orlena Gotel, and Ilka Philippow. Motivation Matters in the Traceability Trenches. In *Proc. 17th International Requirements Engineering Conference (RE09)*, pages 143–148, 2009.

[RMK13] Patrick Rempel, Patrick Mäder, and Tobias Kuschke. An Empirical Study on Project-Specific Traceability Strategies. In *Proc. 21st International Requirements Engineering Conference (RE13)*, pages 195–204, 2013.