Managing technical debt through software metrics, refactoring and traceability

Charalampidou, Sofia

Appendix A

A1. Supplementary Material to Chapter 6 – Primary studies

Alaa, G., & Samir, Z. (2014). A multi-faceted roadmap of requirements traceability types adoption in SCRUM: An empirical study. *9th International Conference on Informatics and Systems*, SW-1-SW-9.

Alhindawi, N., Meqdadi, O., Bartman, B., & Maletic, J. I. (2013). A tracelab-based solution for identifying traceability links using LSI. *7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)*, 79-82.

Ali, N., Sharafi, Z., Guéhéneuc, YG., Antoniol, G. (2015). *Empirical Software Engineering*, 20(2), 442–478.

Amar, B., Leblanc, H., Coulette, B., & Nebut, C. (2010). Using Aspect-Oriented Programming to Trace Imperative Transformations. *14th IEEE International Enterprise Distributed Object Computing Conference*, 143-152.

Antoniol, G., Canfora, G., & De Lucia, A. (1999). Maintaining traceability during object-oriented software evolution: a case study. *Proceedings of the IEEE International Conference on Software Maintenance (ICSM)*, 211-219.

Bavota, G., De Lucia, A., Oliveto, R., & Tortora, G. (2014). Enhancing software artefact traceability recovery processes with link count information. *Information and Software Technology*, 56(2), 163-182.

Bavota, G., De Lucia, A., Oliveto, R., Panichella, A., Ricci, F., & Tortora, G. (2013). The role of artefact corpus in LSI-based traceability recovery. *7th International Work-*

shop on Traceability in Emerging Forms of Software Engineering (TEFSE), 83-89.

Bhansali, S., Chen, W.-K., de Jong, S., Edwards, A., Murray, R., Drinić, M., Mihočka, D., & Chau, J. (2006). Framework for instruction-level tracing and analysis of program executions. *In Proceedings of the 2nd international conference on Virtual execution environments(VEE)*, ACM, New York, NY, USA, 154-163.

Bianchi, A., Fasolino, A. R., &Visaggio, G. (2000). An exploratory case study of the maintenance effectiveness of traceability models. Proceedings of the 8th International Workshop on Program Comprehension (IWPC), 149-158.

Borg, M., & Pfahl, D. (2011). Do better IR tools improve the accuracy of engineers' traceability recovery? *In Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering (MALETS)*, ACM, New York, NY, USA, 27-34.

Borg, M., Wnuk, K., & Pfahl, D. (2012). Industrial Comparability of Student Artifacts in Traceability Recovery Research - An Exploratory Survey, *16th European Conference on Software Maintenance and Reengineering*, 181-190.

Briand, L. C., Labiche, Y., & Leduc, J. (2005). Tracing distributed systems executions using AspectJ. *21st IEEE International Conference on Software Maintenance (ICSM)*, 81-90.

Briand, L. C., Labiche, Y., Yue, T. (2009). Automated traceability analysis for UML model refinements. *Information and Software Technology*, 51(2), 512-527.

Briand, L., Falessi, D., Nejati, S., Sabetzadeh, M., & Yue, T. (2014). Traceability and SysML design slices to support safety inspections: A controlled experiment. *ACM Transactions on Software Engineering Methodoly*, 23(1), Article 9.

Buchmann, R.A. & Karagiannis, D. (2017). Modelling mobile app requirements for semantic traceability. *Requirements Engineering*, 22(1), 41–75.

Bulbun, G., & Shahzada, H. M. A. (2016). BPMN process model checking using traceability. *6th International Conference on Innovative Computing Technology (INTECH)*, 694-699.

Capobianco, G., De Lucia, A., Oliveto, R., Panichella, A., & Panichella, S. (2009). On the role of the nouns in IR-based traceability recovery, *17th IEEE International Conference on Program Comprehension (ICPC)*, 148-157.

Capobianco, G., De Lucia, A., Oliveto, R., Panichella, A., & Panichella, S. (2009). Traceability Recovery Using Numerical Analysis. *16th Working Conference on Reverse Engineering*, 195-204.

Charrada, E. B., Caspar, D., Jeanneret, C., & Glinz, M. (2011). Towards a benchmark for traceability. In Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution (IWPSE-EVOL), ACM, New York, NY, USA, 21-30.

Chen, X., & Grundy, J. (2011). Improving automated documentation to code traceability by combining retrieval techniques. *In Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE Computer Society, Washington, DC, USA, 223-232.

Cleland-Huang, J., Mäder, P., Mirakhorli, M., & Amornborvornwong, S. (2012). Breaking the big-bang practice of traceability: Pushing timely trace recommendations to project stakeholders. *20th IEEE International Requirements Engineering Conference (RE)*, 231-240.

Corley, C. S., Kraft, N. A., Etzkorn, L. H., & Lukins, S. K. (2011). Recovering traceability links between source code and fixed bugs via patch analysis. *In Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)*, ACM, New York, NY, USA, 31-37.

Cornelissen, B., Zaidman, A., & van Deursen, A. (2011). A Controlled Experiment for Program Comprehension through Trace Visualization. *In IEEE Transactions on Software Engineering*, 37(3), 341-355.

Cornu, B., Barr, E. T., Seinturier, L., & Monperrus, M. (2016). Casper: Automatic tracking of null dereferences to inception with causality traces. *Journal of Systems and Software*, 122, 52-62.

Dagenais, B., Breu, S., Warr, F. W., & Robillard, M. P. (2007). Inferring structural patterns for concern traceability in evolving software. *In Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering (ASE)*, ACM, New York, NY, USA, 254-263.

Dasgupta, T., Grechanik, M., Moritz, E., Dit, B., & Poshyvanyk, D. (2013). Enhancing Software Traceability by Automatically Expanding Corpora with Relevant Documentation. *IEEE International Conference on Software Maintenance (ICSM)*, 320-329.

De Lucia, A., Di Penta, M., Oliveto, R., Panichella, A., & Panichella, S. (2013). Applying a smoothing filter to improve IR-based traceability recovery processes: An empirical investigation. *Information and Software Technology*, 55(4), 741-754.

De Lucia, A., Fasano, F., Oliveto, R., & Tortora, G. (2004). Enhancing an artefact management system with traceability recovery features. *In Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM)*, 306-315.

De Lucia, A., Fasano, F., Oliveto, R., & Tortora, G. (2005). ADAMS Re-Trace: a traceability recovery tool. *9th European Conference on Software Maintenance and Reengineering*, 32-41.

De Lucia, A., Fasano, F., Oliveto, R., & Tortora, G. (2006). Can Information Retrieval Techniques Effectively Support Traceability Link Recovery? *In the 14th IEEE International Conference on Program Comprehension (ICPC)*, 307-316.

De Lucia, A., Fasano, F., Oliveto, R., & Tortora, G. (2007). Recovering traceability links in software artifact management systems using information retrieval methods. *ACM Transactions on Software Engineering Methodolies*, 16(4), Article 13.

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

De Lucia, A., Oliveto, R., & Tortora, G. (2009). The role of the coverage analysis during IR-based traceability recovery: A controlled experiment. *IEEE International Conference on Software Maintenance*, 371-380.

De Lucia, A., Oliveto, R., &Tortora, G. (2008). IR-Based Traceability Recovery Processes: An Empirical Comparison of One-Shot and Incremental Processes. *23rd IEEE/ACM International Conference on Automated Software Engineering*, 39-48.

De Lucia, A., Oliveto, R., Zurolo, F., & Di Penta, M. (2006). Improving Comprehensibility of Source Code via Traceability Information: a Controlled Experiment. *14th IEEE International Conference on Program Comprehension (ICPC)*, 317-326.

Delater, A., & Paech, B. (2013). Tracing Requirements and Source Code during Software Development: An Empirical Study, *ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 25-34.

Di, F., & Zhang, M. (2009). An Improving Approach for Recovering Requirements-to-Design Traceability Links, International Conference on Computational Intelligence and

Software Engineering, 1-6.

Diaz, D., Bavota, G., Marcus, A., Oliveto, R., Takahashi, S., & De Lucia, A. (2013). Using code ownership to improve IR-based Traceability Link Recovery. *21st International Conference on Program Comprehension (ICPC)*, 123-132.

Díaz, J., Pérez, J., & Garbajosa J. (2015). A model for tracing variability from features to product-line architectures: a case study in smart grids. *Requirements Engineering*, 20(3), 323-343.

Duan, C. & Cleland-Huang, J. (2007). Clustering support for automated tracing. *In Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering (ASE)*, ACM, New York, NY, USA, 244-253.

Eaddy, M., Aho, A. V., Antoniol, G., & Guéhéneuc, Y. -G. (2008). CERBERUS: Tracing Requirements to Source Code Using Information Retrieval, Dynamic Analysis, and Program Analysis. *16th IEEE International Conference on Program Comprehension*, 53-62.

Egyed, A., Graf, F., & Grunbacher, P. (2010). Effort and Quality of Recovering Requirements-to-Code Traces: Two Exploratory Experiments. *18th IEEE International Requirements Engineering Conference*, 221-230.

Espinoza, A., & Garbajosa, J. (2008). A Proposal for Defining a Set of Basic Items for Project-Specific Traceability Methodologies. *32nd Annual IEEE Software Engineering Workshop*, 175-184.

Figueiredo, E., Galvao, I., Khan, S. S., Garcia, A., Sant'Anna, C., Pimentel, A., Medeiros, A. L., Fernandes, L., Batista, T., Ribeiro, R., van den Broek, P. M., Aksit, M., Zschaler, S., & Moreira, A. (2009). Detecting architecture instabilities with concern traces: An exploratory study. *Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, 261-264.

Fittkau, F., Finke, S., Hasselbring, W., & Waller, J. (2015). Comparing Trace Visualizations for Program Comprehension through Controlled Experiments, *23rd International Conference on Program Comprehension*, IEEE, 266-276.

Gang, Y., Xianjun, L., Zhongwen L., & Jie, Y. (2011). Fault localization with intersection of control-flow based execution traces. *3rd International Conference on Computer Research and Development*, 430-434.

Gayer, S., Herrmann, A., Keuler, T., Riebisch, M., & Antonino, P. O. (2016).

Lightweight Traceability for the Agile Architect, *In Computer*, 49(5), 64-71.

Gethers, M., Oliveto, R., Poshyvanyk, D., & Lucia, A. D. (2011). On integrating orthogonal information retrieval methods to improve traceability recovery. *27th IEEE International Conference on Software Maintenance (ICSM)*, 133-142.

Ghabi, A., & Egyed, A. (2011). Observations on the connectedness between requirements-to-code traces and calling relationships for trace validation. *26th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 416-419.

Ghabi, A., & Egyed, A. (2012). Code patterns for automatically validating requirements-to-code traces. *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, 200-209.

Ghabi, A., & Egyed, A. (2015). Exploiting traceability uncertainty among artifacts and code. *Journal of Systems and Software*, 108, 178-192

Ghanam, Y., & Maurer, F. (2009). Extreme Product Line Engineering: Managing Variability and Traceability via Executable Specifications. *Agile Conference*, 41-48.

Gotel, O., & Finkelstein, A. (1997). Extended requirements traceability: results of an industrial case study. *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (ISRE)*, 169-178.

Grechanik, M., McKinley, K. S., & Perry, D. E. (2007). Recovering and using use-case-diagram-to-source-code traceability links. *In Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (ESEC-FSE)*, ACM, New York, NY, USA, 95-104.

Guo, J., Monaikul, N., Plepel, C., & Cleland-Huang, J. (2014). Towards an intelligent domain-specific traceability solution. *In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering (ASE)*, ACM, New York, NY, USA, 755-766.

Hammad, M., Collard, M.L. & Maletic, J.I. (2011). Automatically identifying changes that impact code-to-design traceability during evolution. *Software Quality Journal,* 19(1), 35–64.

Hayashi, S., Yoshikawa, T., & Saeki, M. (2010). Sentence-to-Code Traceability Recovery with Domain Ontologies. *Asia Pacific Software Engineering Conference*, 385-

394.

Hayes, J.H., Dekhtyar, A., Sundaram, S.K., Ashlee Holbrook, E., Vadlamudi, S., & April, A. (2007). REquirements TRacing On target (RETRO): improving software maintenance through traceability recovery. *Innovations in Systems and Software Engineering,* 3(3), 193–202

Hegedüs, Á., Horváth, Á., Ráth, I., Rizzi Starr, R., & Varró, D. (2016). Query-driven soft traceability links for models. *Software & System Modeling*,15(3), 733–756.

Heindl, M., & Biffl, S. (2005). A case study on value-based requirements tracing. In Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE), ACM, New York, NY, USA, 60-69.

Helming, J., Koegel, M., & Naughton, H. (2009). Towards traceability from project management to system models. *ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, 11-15.

Jaber, K., Sharif, B., & Liu, C. (2013) A Study on the Effect of Traceability Links in Software Maintenance. *In IEEE Access*, vol. 1, 726-741.

Jain, R., Ghaisas, S., & Sureka, A. (2014). SANAYOJAN: a framework for traceability link recovery between use-cases in software requirement specification and regulatory documents. *In Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, ACM, New York, NY, USA, 12-18.

Jalbert, N., & Sen, K. (2010). A trace simplification technique for effective debugging of concurrent programs. *In Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering (FSE)*, ACM, New York, NY, USA, 57-66.

Javed, M. A., & Zdun, U. (2015). On the effects of traceability links in differently sized software systems. *In Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, ACM, New York, NY, USA, , Article 11 .

Javed, M. A., Stevanetic, S., & Zdun, U. (2015). Cost-Effective Traceability Links for Architecture-Level Software Understanding: A Controlled Experiment. *In Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference (ASWEC)*, ACM, New York, NY, USA, 69-73.

Javed, M. A., Stevanetic, S., & Zdun, U. (2016). Towards a pattern language for construction and maintenance of software architecture traceability links. *In Proceedings of the 21st European Conference on Pattern Languages of Programs (EuroPlop)*, ACM, New York, NY, USA, Article 24.

Javed, M. A., & Zdun, U. (2014). The Supportive Effect of Traceability Links in Change Impact Analysis for Evolving Architectures – Two Controlled Experiments. *International Conference on Software and Systems Reuse (ICSR).* Lecture Notes in Computer Science, vol 8919. Springer, Cham

Ji, W., Berger, T., Antkiewicz, M., & Czarnecki, K. (2015). Maintaining feature traceability with embedded annotations. *In Proceedings of the 19th International Conference on Software Product Line (SPLC)*, ACM, New York, NY, USA, 61-70.

Jiang, H., Nguyen, T. N., Chen, I., Jaygarl H., & Chang, C. K. (2008). Incremental Latent Semantic Indexing for Automatic Traceability Link Evolution Management. *23rd IEEE/ACM International Conference on Automated Software Engineering*, 59-68.

Jiang, S., Zhang, H., Wang, Q., & Zhang, Y. (2010). A Debugging Approach for Java Runtime Exceptions Based on Program Slicing and Stack Traces. *10th International Conference on Quality Software*, 393-398.

Jirapanthong, W. & Zisman, A. (2009). XTraQue: traceability for product line systems. *Software & System Modeling*, 8(1), 117–144.

Khan, S. S., & Lock, S. (2009). Concern tracing and change impact analysis: An exploratory study. *ICSE Workshop on Aspect-Oriented Requirements Engineering and Architecture Design*, 44-48.

Kitamura, M., Takagi, M., Yamada, K., & Sasaki, J. (2013). A representation method to simplify traceability links between software artifacts. *12th IEEE International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT)*, 135-140.

Knethen, A. (2001). A trace model for system requirements changes on embedded systems. *In Proceedings of the 4th International Workshop on Principles of Software Evolution (IWPSE)*, ACM, New York, NY, USA, 17-26.

Kong, W., & Hayes, J. H. (2011). Proximity-based traceability: An empirical validation using ranked retrieval and set-based measures. *Workshop on Empirical*

Requirements Engineering (EmpiRE), 45-52.

Krka, I., Brun, Y, & Medvidovic, N. (2014). Automatic mining of specifications from invocation traces and method invariants. *In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*, ACM, New York, NY, USA, 178-189.

Lafet´, R. F., & Maia, M. (2011). An Empirical Assessment of the Use of Execution Traces in Software Maintenance, *25th Brazilian Symposium on Software Engineering*, 154-163.

Li, M., & Liu, S. (2014). Reviewing Formal Specification for Validation Using Animation and Trace Links. *21st Asia-Pacific Software Engineering Conference*, 263-270.

Li, M., & Liu, S. (2014). Traceability-Based Formal Specification Inspection. *8th International Conference on Software Security and Reliability (SERE),* 167-176.

Lin, Y., & Zhou, X. (2012). A Traceability Approach to Constructing Feature Model from Use Case Models. *International Conference on Computer Science and Service System*, 545-548.

Linsbauer, L., Lopez-Herrejon, E. R., & Egyed, A. (2013). Recovering traceability between features and code in product variants. *In Proceedings of the 17th International Software Product Line Conference (SPLC)*, ACM, New York, NY, USA, 131-140.

Liu, D., & Xu, S. (2009). A Tool Suite for Java Program Tracing and Feature Location. 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, 469-474.

Liu, D., Marcus, A., Poshyvanyk, D., & Rajlich, V. (2007). Feature location via information retrieval based filtering of a single scenario execution trace. *In Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering (ASE)*, ACM, New York, NY, USA, 234-243.

Lohar, S., Amornborvornwong, S., Zisman, A., & Cleland-Huang, J. (2013). Improving trace accuracy through data-driven configuration and composition of tracing features. *In Proceedings of the 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, ACM, New York, NY, USA, 378-388.

Lormans, M., & van Deursen, A. (2006). Can LSI help reconstructing requirements traceability in design and test?, *In Conference on Software Maintenance and Reengi-*

neering (CSMR), 10 -56.

Lou, J.-G., Fu, Q., Yang, S., Li, Y., & Wu, B. (2010). Mining program workflow from interleaved traces. *In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, ACM, New York, NY, USA, 613-622.

Mäder, P. & Cleland-Huang, J. A visual language for modeling and executing traceability queries. *Software & System Modeling,* 12(3), 537–553.

Mäder, P., & Egyed, A. (2011). Do software engineers benefit from source code navigation with traceability? — An experiment in software change management. *26th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 444-447.

Mäder, P., & Egyed, A. (2012).Assessing the effect of requirements traceability for software maintenance, *28th IEEE International Conference on Software Maintenance (ICSM)*, 171-180.

Mäder, P., & Egyed, A. (2015). Do developers benefit from requirements traceability when evolving and maintaining a software system? *Empirical Software Engineering,* 20(2), 413-441.

Mäder, P., & Gotel, O. (2012). Towards automated traceability maintenance. *Journal of Systems and Software*, 85(10), 2205-2227.

Mäder, P., Gotel, O., & Philippow, I. (2008). Rule-Based Maintenance of Post-Requirements Traceability Relations. *16th IEEE International Requirements Engineering Conference*, 23-32.

Mader, P., Gotel, O., & Philippow, I. (2009). Motivation Matters in the Traceability Trenches. *17th IEEE International Requirements Engineering Conference*, 143-148.

Mahmoud, A. & Niu, N. (2014). Supporting requirements to code traceability through refactoring. *Requirements Engineering*, 19(3), 309–329.

Mahmoud, A. & Niu, N. (2015). On the role of semantics in automated requirements tracing. *Requirements Engineering*, 20(3), 281–300.

Mahmoud, A., & Niu, N. (2011). Source code indexing for automated tracing. In Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE), ACM, New York, NY, USA, 3-9.

Mahmoud, A., Niu, N., & Xu, S. (2012). A semantic relatedness approach for traceability link recovery. *20th IEEE International Conference on Program Comprehension (ICPC)*, 183-192.

Maia, M. A., & Lafeta, R. F. (2013). On the impact of trace-based feature location in the performance of software maintainers. *Journal of Systems and Software*, 86(4), 1023-1037.

Mao, C., Tervonen, I., & Chen, J. (2011). Diagnosing Web Services System Based on Execution Traces Pattern Analysis. *8th IEEE International Conference on e-Business Engineering*, 207-214.

Marcus, A., & Maletic, J. I. (2003). Recovering documentation-to-source-code traceability links using latent semantic indexing. 25th International Conference on Software Engineering, 125-135.

Maro, S., Anjorin, A., Wohlrab, R., & Steghöfer, J. (2016). Traceability maintenance: Factors and guidelines. *31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 414-425.

Marques, A., Ramalho, F., & Andrade, W. L. (2015). Towards a requirements traceability process centered on the traceability model. *In Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC)*, ACM, New York, NY, USA, 1364-1369.

McMillan, C., Poshyvanyk, D., & Revelle, M. (2009). Combining textual and structural analysis of software artifacts for traceability link recovery, *ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, 41-48.

Mirakhorli, M., & Cleland-Huang, J. (2011). Transforming trace information in architectural documents into re-usable and effective traceability links. *In Proceedings of the 6th International Workshop on SHAring and Reusing Architectural Knowledge (SHARK).* ACM, New York, NY, USA, 45-52.

Mirakhorli, M., & Cleland-Huang, J. (2016). Detecting, Tracing, and Monitoring Architectural Tactics in Code. *In IEEE Transactions on Software Engineering*, 42(3), 205-220.

Mishra, A., & Misra, A. K. (2011). Formal Aspects of Specification and Validation of Dynamic Adaptive System by Analyzing Execution Traces. *8th IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems*, 49-58.

Mohan, K., Xu, P., Cao, L., & Ramesh, B. (2008). Improving change management in software development: Integrating traceability and software configuration management. *Decision Support Systems,* 45(4), 922-936.

Moros, B., Toval, A., Rosique, F., & Sánchez, P. (2013). Transforming and tracing reused requirements models to home automation models. *Information and Software Technology,* 55(6), 941-965.

Nejati, S., Sabetzadeh, M., Falessi, D., Briand, L., & Coq, T. (2012). A SysML-based approach to traceability management and design slicing in support of safety certification: Framework, tool support, and case studies. *Information and Software Technology*, 54(6), 569-590.

Neumuller, C., & Grunbacher, P. (2006). Automating Software Traceability in Very Small Companies: A Case Study and Lessons Learne. 21st IEEE/ACM International Conference on Automated Software Engineering (ASE), 145-156.

Niu, N., Wang, W., & Gupta, A. (2016). Gray links in the use of requirements traceability. *In Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*, ACM, New York, NY, USA, 384-395.

Pandanaboyana, S., Sridharan, S., Yannelli, J., & Hayes, J. H. (2013). REquirements TRacing On target (RETRO) enhanced with an automated thesaurus builder: An empirical study. *7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)*, 61-67.

Panichella, A., De Lucia, A., & Zaidman, A. (2015). Adaptive User Feedback for IR-Based Traceability Recovery. *8th IEEE/ACM International Symposium on Software and Systems Traceability*, 15-21.

Panichella, A., McMillan, C., Moritz, E., Palmieri, D., Oliveto, R., Poshyvanyk, D., & De Lucia, A. (2013). When and How Using Structural Information to Improve IR-Based Traceability Recovery. *17th European Conference on Software Maintenance and Reengineering*, 199-208.

Parizi, R. M. (2016). On the gamification of human-centric traceability tasks in software testing and coding. *14th IEEE International Conference on Software Engineering Research, Management and Applications (SERA)*, 193-200.

Pradel, M., & Gross, T. R. (2009). Automatic Generation of Object Usage Specifications from Large Method Traces. *IEEE/ACM International Conference on Automated Software Engineering*, 371-382.

Pruski, P., Lohar, S., Goss, W. Rasin, A., Cleland-Huang, J. (2015). TiQi: answering unstructured natural language trace queries. *Requirements Engineering*, 20(3), 215–232.

Qusef, A., Bavota, G., Oliveto, R., De Lucia, A., & Binkley, D. (2011). SCOTCH: Test-to-code traceability using slicing and conceptual coupling. *27th IEEE International Conference on Software Maintenance (ICSM),* 63-72.

Qusef, A., Oliveto, R., & De Lucia, A. (2010). Recovering traceability links between unit tests and classes under test: An improved method. *IEEE International Conference on Software Maintenance (ICSM)*, 1-10.

Rafati, A., Peck Lee, S., Parizi, R. M., & Zamani, S. (2015). A test-to-code traceability method using .NET custom attributes. *In Proceedings of the 2015 Conference on research in adaptive and convergent systems (RACS)*, ACM, New York, NY, USA, 489-496.

Rahimi, M., Goss, W., & Cleland-Huang, J. (2016). Evolving Requirements-to-Code Trace Links across Versions of a Software System. *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 99-109.

Regan, G., Flood, D.,& McCaffery, F. (2016). Research Findings from an Industrial Trial of a Traceability Assessment and Implementation Framework. *IEEE/ACM International Conference on Software and System Processes (ICSSP)*, 91-95.

Rempel, P., & Mäder, P. (2015). A quality model for the systematic assessment of requirements traceability. *23rd IEEE International Requirements Engineering Conference (RE)*, 176-185.

Rempel, P., Mäder, P., & Kuschke, T. (2013). Towards feature-aware retrieval of refinement traces. *7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)*, 100-104.

Roehm, T., Nosovic, S., & Bruegge, B. (2015).Automated extraction of failure reproduction steps from user interaction traces. *22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 121-130.

Saiedian, H., Kannenberg, A. & Morozov, S. (2013). A streamlined, cost-effective database approach to manage requirements traceability. *Software Quality Journal*, 21(1), 23-28.

Santos, J. C. S., Mirakhorli, M., Mujhid, I., & Zogaan, W. (2016). BUDGET: A Tool for Supporting Software Architecture Traceability Research. *13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 303-306.

Sardinha, A., Yu, Y., Niu, N., & Rashid, A. (2012). EA-tracer: identifying traceability links between code aspects and early aspects. *In Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC)*, ACM, New York, NY, USA, 1035-1042.

Schroter, A., Schröter, A., Bettenburg, N., & Premraj, R. (2010). Do stack traces help developers fix bugs? *7th IEEE Working Conference on Mining Software Repositories (MSR)*, 118-121.

Sena Marques, M. R., Siegert, E., & Brisolara, L. (2014). Integrating UML, MARTE and sysml to improve requirements specification and traceability in the embedded domain. *12th IEEE International Conference on Industrial Informatics (INDIN)*, 176-181.

Settimi, R., Cleland-Huang, J., Ben Khadra, O., Mody, J., Lukasik, W., & DePalma, C. (2004). Supporting software evolution through dynamically retrieving traces to UML artifacts, *Proceedings of the 7th International Workshop on Principles of Software Evolution*, 49-54.

Shahid, M., & Ibrahim, S. (2016). Change impact analysis with a software traceability approach to support software maintenance. *13th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 391-396.

Shin, Y., & Cleland-Huang, J. (2012). A comparative evaluation of two user feedback techniques for requirements trace retrieval. *In Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC)*, ACM, New York, NY, USA, 1069-1074.

Shin, Y., Hayes, J. H., & Cleland-Huang, J. (2015). Guidelines for Benchmarking Automated Software Traceability Techniques. *8th IEEE/ACM International Symposium on Software and Systems Traceability*, 61-67.

Spanoudakis, G., Zisman, A., Pérez-Miñana, E., & Krause, P. (2004). Rule-based generation of requirements traceability relations. *Journal of Systems and Software*, 72(2), 105-127.

Sundaram, S.K., Hayes, J.H., Dekhtyar, A. et al. (2010). Assessing traceability of software engineering artifacts. *Requirements Engineering*, 15(3), 313–335.

Sureka, A., Lal, S., & Agarwal, L. (2011). Applying Fellegi-Sunter (FS) Model for Traceability Link Recovery between Bug Databases and Version Archives. *18th Asia-Pacific Software Engineering Conference*, 146-153.

Tabares, M. S., Moreira, A., Anaya, R., Arango, F., & Araujo, J. (2007). A Traceability Method for Crosscutting Concerns with Transformation Rules, Early Aspects. *ICSE Workshops in Aspect-Oriented Requirements Engineering and Architecture Design (EARLYASPECTS)*, 7-7.

Taniguchi, K., Ishio, T., Kamiya, T., Kusumoto, S., & Inoue, K. (2005). Extracting sequence diagram from execution trace of Java program. *8th International Workshop on Principles of Software Evolution (IWPSE)*, 148-151.

Toda, T., Kobayashi, T., Atsumi, N., & Agusa, K. (2013). Grouping Objects for Execution Trace Analysis Based on Design Patterns. *20th Asia-Pacific Software Engineering Conference (APSEC)*, 25-30.

von Knethen, A. (2002). Change-oriented requirements traceability. Support for evolution of embedded systems. *In Proceedings of the International Conference on Software Maintenance (ICSM)*, 482-485.

Wang, W., Niu, N., Liu, H., & Wu, Y. (2015). Tagging in Assisted Tracing. *8th IEEE/ACM International Symposium on Software and Systems Traceability*, 8-14.

Wang, X., Gu, Q., Zhang, X., Chen, X., & Chen, D. (2009). Fault Localization Based on Multi-level Similarity of Execution Traces. *16th Asia-Pacific Software Engineering Conference*, 399-405.

Wieloch, M., Amornborvornwong, S., & Cleland-Huang, J. (2013). Trace-by-classification: A machine learning approach to generate trace links for frequently occurring software artifacts. *7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)*, 110-114.

Wohlrab, R., Steghöfer, J., Knauss, E., Maro, S., & Anjorin, A. (2016). Collaborative Traceability Management: Challenges and Opportunities. *24th IEEE International Requirements Engineering Conference (RE)*, 216-225.

Wong, C., Xiong, Y., Zhang, H., Hao, D., Zhang, L., & Mei, H. (2014). Boosting Bug-Report-Oriented Fault Localization with Segmentation and Stack-Trace Analysis. *IEEE*

International Conference on Software Maintenance and Evolution, 181-190.

Yamada, S., Ugumori, M., & Kusumoto, S. (2010). A Software Tag Generation System to Realize Software Traceability. *Asia Pacific Software Engineering Conference*, 423-432

Yoshikawa, T., Hayashi, S., & Saeki, M. (2009). Recovering traceability links between a simple natural language sentence and source code using domain ontologies, *IEEE International Conference on Software Maintenance*, 551-554.

Zhang, H., Jiang, S., & Jin, R. (2011). An improved static program slicing algorithm using stack trace. *2nd IEEE International Conference on Software Engineering and Service Science*, 563-567.

Zhou, X., Huo, Z., Huang, Y., & Xu, J. (2008). Facilitating Software Traceability Understanding with ENVISION. *32nd Annual IEEE International Computer Software and Applications Conference*, Turku, 295-302.

Ziftci, C., & Krüger, I. (2013). Test intents: enhancing the semantics of requirements traceability links in test cases. *In Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC)*, ACM, New York, NY, USA, 1272-1277.

Zou, X., Settimi, R. & Cleland-Huang, J. (2010). Improving automated requirements trace retrieval: a study of term-based enhancement methods. *Empirical Software Engineering*, 15(2), 119–146.

Zou, X., Settimi, R., & Cleland-Huang, J. (2006). Phrasing in Dynamic Requirements Trace Retrieva. *30th Annual International Computer Software and Applications Conference (COMPSAC)*, 265-272.

A2. Supplementary Material to Chapter 6 – Additional Data for Research Questions

RQ1: Detailed presentation of connected artifacts (and the respective phases they belong to)

The tables below present further information about the top-5 most frequently traced software artifact types. Specifically, there is one table for each artifact, which shows the count of studies in which this artifact has been linked with other types of artifacts (as well as the development phases these artifacts belong to). We note that the tables present only pairs that have been found in at least 5 studies.

Table A2.1: Count of studies connecting Requirements to other artifacts

Artifact 1	Artifact 2	Development Phases		Count
Requirements	Source Code	R	I	21
	Classes		I	14
	Test Cases		T	10
	Methods		I	5
	Design Models		D	5
	Requirements		R	5

Table A2.2: Count of studies connecting Source Code (in general) to other artifacts

Artifact 1	Artifact 2	Development Phases		Count
Source Code	Requirements	I	R	21
	Test Cases		T	7
	Specifications		-	5
	Features		R	5
	Design Models		D	4
	UML Diagrams		D	4

Table A2. 3: Count of studies connecting Classes to other artifacts

Artifact 1	Artifact 2	Development Phases		Count
Classes	Use Cases	I	R	15
	Requirements		R	14
	Test Cases		T	10
	Interaction Diagrams		D	6
	Features		R	4

Table A2.4: Count of studies connecting UML diagrams to other artifacts

Artifact 1	Artifact 2	Development Phases		Count
UML Diagrams	Source code	D	I	4
	Requirements		R	2
	Use Cases		R	2
	Classes		I	2

Table A2. 5: Count of studies connecting Use Cases to other artifacts

Artifact 1	Artifact 2	Development Phases		Count
Use Cases	Classes	D	I	15
	Interaction Diagrams		D	6
	Test Cases		T	6
	Source code		I	4
	Requirements		R	3
	Features		R	3
	Methods		I	3

RQ4: View on the development phases and the exact artifacts being examined by using different research methods

Table A2. 6 below shows the top-5 (when applicable) pairs of development phases studied by using each empirical research method

Table A2. 6: Pairs of development phases studied using the different research methods

Research Method	**Development Phases**	**Count**
Case study	R-I	60
	R-R	30
	R-D	26
	I-T	23
	D-I	20
Experiment	R-I	48
	D-I	40
	R-D	31
	D-D	23
	I-I	13
Proof of Concept	R-I	8
	D-I	8
	R-D	8
	I-T	6
	I-I	5
Survey	R-D	1
	R-R	1
	R-I	1
	R-T	1
	I-T	1
Simulation	R-R	2
	R-I	1

Table A2.7 shows the most frequently traced pairs of software artifacts and how they are distributed based on the empirical research method used when studied.

Table A2.7: Research methods used for studding the most frequently traced pairs of software artifacts

Artifact 1	**Artifact 2**	**Case Study**	**Experiment**	**Proof of Concept**	**Survey**
Requirements	Source Code	12	5	4	1
Use Cases	Classes	9	6		
Requirements	Classes	6	7	1	
Classes	Test Cases	6	3	1	
Requirements	Test Cases	6	2	1	1
Source Code	Test Cases	7			1
Interaction Diagrams	Test Cases	4	2		
Interaction Diagrams	Classes	3	4		
Use Cases	Test Cases	3	3		
Use Cases	Interaction Diagrams	3	4		
High Level Require-	Low Level Requirements	5	1		
Source Code	Specifications	3	2		
Features	Source Code	4		1	
Requirements	Methods	2	3		
Requirements	Design Models	2	2		1