References

3SL. (2018). Retrieved July 6, 2017, from https://www.threesl.com/cradle/

Acharya, M., & Robinson, B. (2011). Practical change impact analysis based on static program slicing for industrial software systems. In *33rd International Conference on Software Engineering - ICSE '11* (pp. 746–755). New York, USA: ACM. https://doi.org/10.1145/1985793.1985898

AjaxSwing. (2018). Retrieved October 19, 2018, from http://creamtec.com/products/ajaxswing/index.html

Alves-Foss, J., Conte de Leon, D., & Oman, P. (2002). Experiments in the use of XML to enhance traceability between object-oriented design specifications and source code. In *35th Annual Hawaii International Conference on System Sciences* (pp. 3959–3966). IEEE. https://doi.org/10.1109/HICSS.2002.994466

Apiwattanapong, T., Orso, A., & Harrold, M. J. (2004). A differencing algorithm for object-oriented programs. In *19th International Conference on Automated Software Engineering, 2004.* (pp. 2–13). IEEE. https://doi.org/10.1109/ASE.2004.1342719

Apiwattanapong, T., Orso, A., & Harrold, M. J. (2005). Efficient and precise dynamic impact analysis using execute-after sequences. In *27th International Conference on Software Engineering - ICSE '05* (pp. 432–441). New York, USA: ACM. https://doi.org/10.1145/1062455.1062534

ArchStudio. (2018). Retrieved July 5, 2017, from http://isr.uci.edu/projects/archstudio/setup-easy.html

Arunthavanathan, A., Shanmugathasan, S., Ratnavel, S., Thiyagarajah, V., Perera, I., Meedeniya, D., & Balasubramaniam, D. (2016). Support for traceability management of software artefacts using Natural Language Processing. In *Moratuwa Engineering Research Conference (MERCon)* (pp. 18–23). IEEE. https://doi.org/10.1109/MERCon.2016.7480109

Athira, B., & Samuel, P. (2011). Traceability Matrix for Regression Testing in Distributed Software Development. In *Abraham A., Lloret M. J., Buford J. F., Suzuki J., Thampi S. M. (eds) Advances in Computing and Communications. ACC 2011*. Communications in Computer and Information Science, 191, (pp. 80–87). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-22714-1_9

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual SUS scores mean: adding an adjective rating scale. *Journal of Usability Studies*, *4*(3), 114–123.

Bass, L. J., Weber, I. M., & Zhu, L. (2015). *DevOps : a software architect's perspective* (1st ed.). Addison-Wesley Professional.

Bavota, G., Colangelo, L., De Lucia, A., Fusco, S., Oliveto, R., & Panichella, A. (2012). TraceME: Traceability Management in Eclipse. In *28th IEEE International Conference on Software Maintenance (ICSM)* (pp. 642–645). IEEE. https://doi.org/10.1109/ICSM.2012.6405343

Berg, A. M. (2015). *Jenkins Continuous Integration Cookbook* (2nd ed.). Packt Publishing.

Bitrix24. (2018). Retrieved October 2, 2018, from https://www.bitrix24.com/

Borg, M., Wnuk, K., Regnell, B., & Runeson, P. (2017). Supporting Change Impact Analysis Using a Recommendation System: An Industrial Case Study in a Safety-Critical Context. *IEEE Transactions on Software Engineering*, *43*(7), 675–700. https://doi.org/10.1109/TSE.2016.2620458

Borgatti, S. P. (2005). Centrality and network flow. *Social Networks*, *27*(1), 55–71. https://doi.org/10.1016/j.socnet.2004.11.008

Borland. (2006). *Borland ® CaliberRM ™*.

Brooke, J. (2013). SUS: A Retrospective. *Journal of Usability Studies*, *8*(2), 29–40.

Burgaud, L. (2006). A Novel development framework combining requirement driven and model based engineering processes. In *4ème Conférence Annuelle d'Ingénierie Système* (pp. 1–7).

Chang, S. K. (2005). *Handbook of Software Engineering And Knowledge Engineering: Recent Advances*. *World Scientific Publishing*. World Scientific Publishing. https://doi.org/10.1142/5785

Chawathe, S. S., Rajaraman, A., Garcia-Molina, H., & Widom, J. (1996). Change detection in hierarchically structured information. *ACM SIGMOD Record*, *25*(2), 493–504. https://doi.org/10.1145/235968.233366

Chen, X., Hosking, J., & Grundy, J. (2012). Visualizing traceability links between source code and documentation. In *IEEE Symposium on Visual Languages and Human-Centric Computing*

(VL/HCC) (pp. 119–126). IEEE. https://doi.org/10.1109/VLHCC.2012.6344496

Cleland-Huang, J., Chang, C. K., & Christensen, M. (2003). Event-based traceability for managing evolutionary change. *IEEE Transactions on Software Engineering*, *29*(9), 796–810. https://doi.org/10.1109/TSE.2003.1232285

Cleland-Huang, J., Gotel, O. C. Z., Hayes, J. H., Mäder, P., & Zisman, A. (2014). Software traceability: trends and future directions. In *Future of Software Engineering - FOSE 2014* (pp. 55–69). New York, USA: ACM. https://doi.org/10.1145/2593882.2593891

Cleland-Huang, J., Zisman, A., & Gotel, O. (2012). *Software and Systems Traceability*. *Software and Systems Traceability* (1st ed.). London: Springer-Verlag London. https://doi.org/10.1007/978-1-4471-2239-5

Cobena, G., Abiteboul, S., & Marian, A. (2002). Detecting changes in XML documents. In *18th International Conference on Data Engineering* (pp. 41–52). IEEE. https://doi.org/10.1109/ICDE.2002.994696

Czibula, I. G., Czibula, G., Miholca, D. L., & Marian, Z. (2017). Identifying Hidden Dependencies in Software Systems. *Studia Universitatis Babeș-Bolyai Informatica*, *62*(1), 90–106. https://doi.org/10.24193/subbi.2017.1.07

D3.js. (2018). Retrieved May 21, 2018, from http://d3js.org/

Dantas, C., Murta, L., & Werner, C. (2007). Mining Change Traces from Versioned UML Repositories. In *Brazilian Symposium on Software Engineering (SBES'07)* (pp. 236–252).

De Lucia, A., Fasano, F., & Oliveto, R. (2008). Traceability management for impact analysis. In *Frontiers of Software Maintenance* (pp. 21–30). IEEE. https://doi.org/10.1109/FOSM.2008.4659245

De Lucia, A., Oliveto, R., & Tortora, G. (2008). Adams re-trace: traceability link recovery via latent semantic indexing. In *13th International Conference on Software Engineering - ICSE '08* (pp. 839–842). New York, USA: ACM. https://doi.org/10.1145/1368088.1368216

Déhoulé, F., Badri, L., & Badri, M. (2017). A Change Impact Analysis Model for Aspect Oriented Programs. In *12th International Conference on Evaluation of Novel Approaches to Software Engineering* (pp. 144–157). SCITEPRESS Publications. https://doi.org/10.5220/0006350701440157

diffmk. (2018). Retrieved June 27, 2018, from https://docs.oracle.com/cd/E36784_01/html/E36870/diffmk-1.html#scrolltoc

Docker. (2018). Retrieved August 28, 2018, from https://docs.docker.com

Duarte, A. M. D., Duarte, D., & Thiry, M. (2016). TraceBoK: Toward a Software Requirements Traceability Body of Knowledge. In *24th International Requirements Engineering Conference (RE)* (pp. 236–245). IEEE. https://doi.org/10.1109/RE.2016.32

Duvall, P. M., Matyas, S., & Glover, A. (2007). *Continuous Integration: Improving Software Quality and Reducing Risk*. *Addison-Wesley Professional* (1st ed.). Addison-Wesley Professional.

DZone DevOps. (2018). Retrieved August 28, 2018, from https://dzone.com/articles/what-is-cicd

Eck, A., Uebernickel, F., & Brenner, W. (2014). Fit for continuous integration: how organizations assimilate an agile practice. In *20th Americas Conference on Information Systems (AMCIS '14)* (pp. 1–11). GA, USA: AIS. https://doi.org/10.1109/ACCESS.2017.2685629

Egyed, A. (2001). A scenario-driven approach to traceability. In *23rd International Conference on Software Engineering. ICSE 2001* (pp. 123–132). IEEE. https://doi.org/10.1109/ICSE.2001.919087

Farcic, V. (2016). *The DevOps 2.0 Toolkit: Automating the Continuous Deployment Pipeline with Containerized Microservices* (1st ed.). CreateSpace Independent Publishing Platform.

Faulkner, L. (2003). Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, *35*(3), 379–383. https://doi.org/10.3758/BF03195514

Fernández, P. (2008). Book Review: Google's PageRank and Beyond: The Science of Search Engine Rankings. *The Mathematical Intelligencer*, 30(1), 68–69. https://doi.org/10.1007/BF02985759

Filho, G. A. de A. C., & Lencastre, M. (2012). Towards a Traceability Visualisation Tool. In *8th International Conference on the Quality of Information and Communications Technology* (pp. 221–223). IEEE. https://doi.org/10.1109/QUATIC.2012.60

Fockel, M., Holtmann, J., & Meyer, J. (2012). Semi-automatic establishment and maintenance of valid traceability in automotive development processes. In *2nd International Workshop on Software Engineering for Embedded Systems (SEES)* (pp. 37–43). IEEE. https://doi.org/10.1109/SEES.2012.6225489

Galbo, S. P. D. (2010). *A Survey of Impact Analysis Tools for Effective Code Evolution*. University of Central Florida.

Galvão, I., & Goknil, A. (2007). Survey of traceability approaches in model-driven engineering. In *IEEE International Enterprise Distributed Object Computing Workshop, EDOC* (pp. 313–324). IEEE. https://doi.org/10.1109/EDOC.2007.4384003

Gephi. (2017). Retrieved October 14, 2017, from https://gephi.org/

Ghantous, G. B., & Gill, A. (2017). DevOps: Concepts, Practices, Tools, Benefits and Challenges. In *21st Pacific Asia Conference on Information Systems* (pp. 1–13). Langkawi, Malaysia: AIS.

GNU "ed." (2018). Retrieved June 27, 2018, from http://www.gnu.org/software/ed/manual/ed_manual.html

Goknil, A., Kurtev, I., & Berg, K. van den. (2016). A Rule-Based Change Impact Analysis Approach in Software Architecture for Requirements Changes. *Eprint ArXiv:1608.02757*, 1–44.

Goknil, A., Kurtev, I., van den Berg, K., & Spijkerman, W. (2014). Change impact analysis for requirements: A metamodeling approach. *Information and Software Technology*, *56*(8), 950–972. https://doi.org/10.1016/j.infsof.2014.03.002

Graph Visualization-Neo4j. (2018). Retrieved July 21, 2017, from https://neo4j.com/developer/guide-data-visualization/

Hambling, B., & Goethem, P. van. (2013). *User Acceptance Testing: A Step-by-step Guide* (1st ed.). BCS, The Chartered Institute for IT.

Hattori, L., Guerrero, D., Figueiredo, J., Brunet, J., & Dam, J. (2008). On the Precision and Accuracy of Impact Analysis Techniques. In *7th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008)* (pp. 513–518). IEEE. https://doi.org/10.1109/ICIS.2008.104

Hayes, J. H., Dekhtyar, A., Sundaram, S. K., Holbrook, E. A., 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. https://doi.org/10.1007/s11334-007-0024-1

Herman, I., Melancon, G., & Marshall, M. S. (2000). Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, *6*(1), 24–43. https://doi.org/10.1109/2945.841119

Holten, D. (2006). Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, *12*(5), 741–748. https://doi.org/10.1109/TVCG.2006.147

IBM-Rational DOORS. (2017). Retrieved October 14, 2017, from https://www.ibm.com/us-en/marketplace/rational-doors

Ibrahim, S., Idris, N. B., Munro, M., & Deraman, A. (2005). Integrating Software Traceability for Change Impact Analysis. *Integrating Software Traceability for Change Impact Analysis*, *2*(4), 301–308.

Ibrahim, S., Munro, M., & Deraman, A. (2005). A Requirements Traceability To Support Change Impact Analysis. *Asian Journal of Information Technology*, *4*(4), 335–344.

Jaro Winkler Distance. (2017). Retrieved October 14, 2017, from http://alias-i.com/lingpipe/docs/api/com/aliasi/spell/JaroWinklerDistance.html

Jashki, M.-A., Zafarani, R., & Bagheri, E. (2008). Towards a more efficient static software change impact analysis method. In *8th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering - PASTE '08* (pp. 84–90). New York, USA: ACM. https://doi.org/10.1145/1512475.1512493

Javed, M. A., & Zdun, U. (2014). A systematic literature review of traceability approaches between software architecture and source code. In *18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14* (pp. 1–10). New York, USA: ACM. https://doi.org/10.1145/2601248.2601278

JFreeChart. (2018). Retrieved August 14, 2018, from http://www.jfree.org/jfreechart/

JIRA Software. (2018). Retrieved October 2, 2018, from https://www.atlassian.com/software/jira

Kabeer, S. J., Nayebi, M., Ruhe, G., Carlson, C., & Chew, F. (2017). Predicting the Vector Impact of Change - An Industrial Case Study at Brightsquid. In *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (pp. 131–140). IEEE. https://doi.org/10.1109/ESEM.2017.20

Kama, N. (2013). Change impact analysis for the software development phase: State-of-the-art. *International Journal of Software Engineering & Its Applications*, *7*(2), 235–244.

Kamalabalan, K., Uruththirakodeeswaran, T., Thiyagalingam, G., Wijesinghe, D. B., Perera, I., Meedeniya, D., & Balasubramaniam, D. (2015). Tool support for traceability of software artefacts. In *Moratuwa Engineering Research Conference (MERCon)* (pp. 318–323). IEEE. https://doi.org/10.1109/MERCon.2015.7112366

Kchaou, D., Bouassida, N., & Ben-Abdallah, H. (2017). UML models change impact analysis using a text similarity technique. *IET Software*, *11*(1), 27–37. https://doi.org/10.1049/iet-sen.2015.0113

Keenan, E., Czauderna, A., Leach, G., Cleland-Huang, J., Shin, Y., Moritz, E., … Hearn, D. (2012). TraceLab: An experimental workbench for equipping researchers to innovate, synthesize, and comparatively evaluate traceability solutions. In *34th International Conference on Software Engineering (ICSE)* (pp. 1375–1378). Zurich, Switzerland: IEEE. https://doi.org/10.1109/ICSE.2012.6227244

Keiningham, T. L., Aksoy, L., Cooil, B., Andreassen, T. W., & Williams, L. (2008). A holistic examination of Net Promoter. *Journal of Database Marketing & Customer Strategy Management*, *15*(2), 79–90. https://doi.org/10.1057/dbm.2008.4

Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. (2016). *The DevOps Handbook* (1st ed.). IT Revolution Press.

Kitsu, E., Omori, T., & Maruyama, K. (2013). Detecting Program Changes from Edit History of Source Code. In *20th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 299–306). IEEE. https://doi.org/10.1109/APSEC.2013.48

Knoke, D., & Yang, S. (2008). *Social Network Analysis* (3rd ed.). London: SAGE Publications.

Kugele, S., & Antkowiak, D. (2016). Visualization of Trace Links and Change Impact Analysis. In *IEEE 24th International Requirements Engineering Conference Workshops (REW)* (pp. 165–169). Beijing, China: IEEE. https://doi.org/10.1109/REW.2016.039

LDRA. (2018). Retrieved July 5, 2017, from http://www.ldra.com/en/software-quality-test-tools/group/by-software-life-cycle/requirements-traceability

Lee, C., Guadagno, L., & Jia, X. (2003). An agile approach to capturing requirements and traceability. In *2nd International Workshop on Traceability in Emerging Forms of Software Engineering* (pp. 1–7). Citeseer.

Lee, J., Cho, B., Youn, H., & Lee, E. (2009). Reliability analysis method for supporting traceability using UML. In *Communications in Computer and Information Science* (pp. 94–101). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-10619-4_12

Lee, M., & Offutt, A. J. (2002). Algorithmic analysis of the impacts of changes to object-oriented software. In *Technology of Object-Oriented Languages and Systems* (pp. 61–70). CA, USA: IEEE. https://doi.org/10.1109/TOOLS.2000.868959

Lee, W.-T., Deng, W.-Y., Lee, J., & Lee, S.-J. (2010). Change impact analysis with a goal-driven traceability-based approach. *International Journal of Intelligent Systems*, *25*(8), 878–908. https://doi.org/10.1002/int.20443

Lehnert, S. (2011). A taxonomy for software change impact analysis. In *12th International Workshop and the 7th Annual ERCIM Workshop on Principles on Software Evolution and Software Evolution - IWPSE-EVOL '11* (pp. 41–50). New York, USA: ACM. https://doi.org/10.1145/2024445.2024454

Lehnert, S. (2015). *Multiperspective Change Impact Analysis to Support Software Maintenance and Reengineering*. University of Hamburg.

Lehnert, S., Farooq, Q. U. A., & Riebisch, M. (2013). Rule-based impact analysis for heterogeneous software artifacts. In *European Conference on Software Maintenance and Reengineering, CSMR* (pp. 209–218). Genova, Italy: IEEE. https://doi.org/10.1109/CSMR.2013.30

Levenshtein-Algorithm. (2017). Retrieved October 14, 2017, from http://www.levenshtein.net/

Li, B., Sun, X., Leung, H., & Zhang, S. (2013). A survey of code-based change impact analysis techniques. *Software Testing Verification and Reliability*, *23*(8), 613–646. https://doi.org/10.1002/stvr.1475

Lindholm, T., Kangasharju, J., & Tarkoma, S. (2006). Fast and simple XML tree differencing by sequence alignment. In *ACM Symposium on Document Engineering - DocEng '06* (pp. 75–84). New York, USA: ACM. https://doi.org/10.1145/1166160.1166183

Lucia, A. De, Fasano, F., Oliveto, R., & Tortora, G. (2007). Recovering traceability links in software artifact management systems using information retrieval methods. *ACM Transactions on Software Engineering and Methodology*, *16*(4), 13:1-13:50. https://doi.org/10.1145/1276933.1276934

Mäder, P., & Gotel, O. (2012). Towards automated traceability maintenance. *Journal of Systems and Software*, *85*(10), 2205–2227. https://doi.org/10.1016/j.jss.2011.10.023

Mäder, P., Gotel, O., Kuschke, T., & Philippow, I. (2008). traceMaintainer - Automated Traceability Maintenance. In *16th IEEE International Requirements Engineering Conference* (pp. 329–330). Catalunya, Spain: IEEE. https://doi.org/10.1109/RE.2008.25

Marcus, A., Xie, X., & Poshyvanyk, D. (2005). When and how to visualize traceability links? In *3rd International Workshop on Traceability in Emerging Forms of Software Engineering - TEFSE '05* (pp. 56–61). New York, USA: ACM. https://doi.org/10.1145/1107656.1107669

Maro, S., Anjorin, A., Wohlrab, R., & Steghöfer, J.-P. (2016). Traceability maintenance: factors and guidelines. In *31st IEEE/ACM International Conference on Automated Software Engineering - ASE 2016* (pp. 414–425). New York, USA: ACM. https://doi.org/10.1145/2970276.2970314

Matplotlib. (2018). Retrieved August 14, 2018, from http://matplotlib.sourceforge.net

Maule, A., Emmerich, W., & Rosenblum, D. S. (2008). Impact analysis of database schema changes. In *13th International Conference on Software Engineering - ICSE '08* (pp. 451–460). New York, USA: ACM. https://doi.org/10.1145/1368088.1368150

Measuring Requirements. (2018). Retrieved August 31, 2018, from https://www.jamasoftware.com/blog/measuring-requirements-product-size-requirements-quality/

MeasuringU. (2018). Retrieved November 7, 2018, from https://measuringu.com/nps-sus/

Mens, T., Buckley, J., Zenger, M., & Rashid, A. (2005). Towards a Taxonomy of Software Evolution. *J. Softw. Maint. Evol.*, *17*(5), 309–332. https://doi.org/10.1002/smr.v17:5

Merten, T., Jüppner, D., & Delater, A. (2011). Improved representation of traceability links in requirements engineering knowledge using Sunburst and Netmap visualizations. In *4th International Workshop on Managing Requirements Knowledge, MaRK'11 - Part of the 19th IEEE International Requirements Engineering Conference, RE'11* (pp. 17–21). Trento, Italy: IEEE. https://doi.org/10.1109/MARK.2011.6046557

Meyer, M. (2014). Continuous integration and its tools. *IEEE Software*, *31*(3), 14–16. https://doi.org/10.1109/MS.2014.58

Mills, C. (2017). Towards the automatic classification of traceability links. In *32nd IEEE/ACM International Conference on Automated Software Engineering-ASE 2017* (pp. 1018–1021). IL, USA: IEEE. https://doi.org/10.1109/ASE.2017.8115723

Mischler, A., & Monperrus, M. (2014). An Approach for Discovering Traceability Links between Regulatory Documents and Source Code Through User-Interface Labels. *Eprint ArXiv:1403.2639*, 1–14.

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. https://doi.org/10.1016/j.dss.2008.03.003

Molhado Project. (2017). Retrieved July 5, 2017, from http://www.ece.iastate.edu/~tien/molhado/index.html

NetworkX. (2018). Retrieved August 19, 2018, from https://networkx.github.io/

Newman, M. (2010). *Networks: An Introduction* (1st ed.). Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199206650.001.0001

Nguyen, T. N., Munson, E. V., & Boyland, J. T. (2004). The molhado hypertext versioning system. In *15th ACM Conference on Hypertext & Hypermedia - HYPERTEXT '04* (pp. 185–194). New York, USA: ACM. https://doi.org/10.1145/1012807.1012859

Nistor, E. C., Erenkrantz, J. R., Hendrickson, S. A., & van der Hoek, A. (2005). ArchEvol: versioning architectural-implementation relationships. In *12th International Workshop on Software Configuration Management - SCM '05* (pp. 99–111). New York, USA: ACM. https://doi.org/10.1145/1109128.1109136

Oliva, G. A., Gerosa, M. A., Milojicic, D., & Smith, V. (2013). A change impact analysis approach for workflow repository management. In *IEEE 20th International Conference on Web Services, ICWS 2013* (pp. 308–315). CA, USA: IEEE. https://doi.org/10.1109/ICWS.2013.49

Olsson, M. (2015). Document Object Model. In *JavaScript Quick Syntax Reference* (pp. 39–44). Berkeley, CA: Springer. https://doi.org/10.1007/978-1-4302-6494-1_10

Omori, T., & Maruyama, K. (2008). A change-aware development environment by recording editing operations of source code. In *International Workshop on Mining Software Repositories - MSR '08* (pp. 31–34). New York, USA: ACM. https://doi.org/10.1145/1370750.1370758

Passos, L., Apel, S., Kästner, C., Czarnecki, K., Wasowski, A., & Guo, J. (2013). Feature Oriented Software Evolution. In *7th International Workshop on Variability Modelling of Software-intensive Systems-VaMoS '13* (pp. 17:1-17:8). Pisa, Italy: ACM. https://doi.org/10.1145/2430502.2430526

Perera, I., Miller, A., & Allison, C. (2017). A Case Study in User Support for Managing OpenSim Based Multi User Learning Environments. *IEEE Transactions on Learning Technologies*, *10*(3), 342–354. https://doi.org/10.1109/TLT.2016.2632126

Pete, I., & Balasubramaniam, D. (2015). Handling the differential evolution of software artefacts: A framework for consistency management. In *IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)* (pp. 599–600). QC, Canada: IEEE. https://doi.org/10.1109/SANER.2015.7081889

Phetmanee, S., & Suwannasart, T. (2014). A Tool for Impact Analysis of Test Cases Based on Changes of a Web Application. In *International MultiConference of Engineers and Computer Scientists-IMECS '14*, I, (pp. 497–500). Hong Kong: IAENG.

Priority Blocking Queue. (2018). Retrieved June 29, 2018, from http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/PriorityBlockingQueue.html

QASource DevOps Experts. (2018). Retrieved August 28, 2018, from https://www.qasource.com/devops#!devops-expertise

Rajlich, V. (2014). Software evolution and maintenance. In *Future of Software Engineering - FOSE 2014* (pp. 133–144). New York, USA: ACM. https://doi.org/10.1145/2593882.2593893

Rational RequisitePro. (2017). Retrieved July 5, 2017, from https://www.oit.va.gov/Services/TRM/ToolPage.aspx?tid=41

Redmiles, D., Van Der Hoek, A., Al-Ani, B., Hildenbrand, T., Quirk, S., Sarma, A., … Trainer, E. (2007). Continuous Coordination: A New Paradigm to Support Globally Distributed Software Development Projects. *Wirtschaftsinformatik*, *49*, 28–38. https://doi.org/10.1038/leu.2008.246

Rempel, P., & Mader, P. (2017). Preventing defects: The impact of requirements traceability completeness on software quality. *IEEE Transactions on Software Engineering*, *43*(8), 777–797. https://doi.org/10.1109/TSE.2016.2622264

Ren, X., Ryder, B. G., Stoerzer, M., & Tip, F. (2005). Chianti: a change impact analysis tool for java programs. In *27th International Conference on Software Engineering - ICSE '05* (pp. 664–665). New York, USA: ACM. https://doi.org/10.1145/1062455.1062598

ReqView. (2017). Retrieved May 7, 2018, from https://www.reqview.com/

Riebisch, M., Bode, S., Farooq, Q. U. A., & Lehnert, S. (2011). Towards comprehensive modelling by inter-model links using an integrating repository. In *18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, ECBS 2011* (pp. 284–291). NV, USA: IEEE. https://doi.org/10.1109/ECBS.2011.32

Rodrigues, A., Lencastre, M., & Filho, G. A. de A. C. (2016). Multi-VisioTrace: Traceability Visualization Tool. In *10th International Conference on the Quality of Information and Communications Technology (QUATIC)* (pp. 61–66). Lisbon, Portugal: IEEE. https://doi.org/10.1109/QUATIC.2016.019

Rubasinghe, I. D., Meedeniya, D. A., & Perera, I. (2017). Towards Traceability Management in Continuous Integration with SAT-analyzer. In *3rd International Conference on Communication*

and Information Processing (ICCIP 2017) (pp. 77–81). Tokyo, Japan: ACM. https://doi.org/10.1145/3162957.3162985

Rubasinghe, I. D., Meedeniya, D. A., & Perera, I. (2018a). Automated Inter-artefact Traceability Establishment for DevOps Practice. In *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS 2018)* (pp. 211–216). Singapore: IEEE. https://doi.org/10.1109/ICIS.2018.8466414

Rubasinghe, I. D., Meedeniya, D. A., & Perera, I. (2018b). Software Artefact Traceability Analyser : A Case-Study on POS System. In *6th International Conference on Communications and Broadband Networking (ICCBN 2018)* (pp. 1–5). Singapore: ACM. https://doi.org/10.1145/3193092.3193094

Rubasinghe, I. D., Meedeniya, D. A., & Perera, I. (2018). Traceability Management with Impact Analysis in DevOps based Software Development. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 1956–1962). Bangalore, India: IEEE. https://doi.org/10.1109/ICACCI.2018.8554399

Sager, T., Bernstein, A., Pinzger, M., & Kiefer, C. (2006). Detecting similar Java classes using tree algorithms. In *International Workshop on Mining Software Repositories - MSR '06* (pp. 65–71). Shanghai, China: ACM. https://doi.org/10.1145/1137983.1138000

Santiago, I., Vara, J. M., De Castro, V., & Marcos, E. (2014). Visualizing Traceability Information with iTrace. In *9th International Conference on Evaluation of Novel Approaches to Software Engineering* (pp. 5–15). SCITEPRESS Publications. https://doi.org/10.5220/0004865400050015

Sarma, A., Redmiles, D. F., & Van Der Hoek, A. (2012). Palantír: Early detection of development conflicts arising from parallel code changes. *IEEE Transactions on Software Engineering*, *38*(4), 889–908. https://doi.org/10.1109/TSE.2011.64

SAT-Analyser. (2018). Retrieved November 10, 2018, from https://sites.google.com/cse.mrt.ac.lk/sat-analyser/case-studies

Selenium. (2018). Retrieved October 2, 2017, from http://www.seleniumhq.org

SERG :ReqAnalyst. (2017). Retrieved July 5, 2017, from http://swerl.tudelft.nl/bin/view/Main/ReqAnalyst

Shahid, M., & Ibrahim, S. (2016). Change impact analysis with a software traceability approach to support software maintenance. In *13th International Bhurban Conference on Applied Sciences and Technology (IBCAST)* (pp. 391–396). IEEE. https://doi.org/10.1109/IBCAST.2016.7429908

Sharafat, A. R., & Tahvildari, L. (2007). A Probabilistic Approach to Predict Changes in Object-Oriented Software Systems. In *11th European Conference on Software Maintenance and Reengineering (CSMR'07)* (pp. 27–38). Amsterdam, Netherlands: IEEE. https://doi.org/10.1109/CSMR.2007.9

Shneiderman, B. (1992). Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics (TOG)*, *11*(1), 92–99. https://doi.org/10.1145/102377.115768

Sinclair, J., & Cardew-Hall, M. (2008). The folksonomy tag cloud: when is it useful? *Journal of Information Science*, *34*(1), 15–29. https://doi.org/10.1177/0165551506078083

Slack. (2018). Retrieved October 2, 2018, from https://slack.com

Sommerville, I. (2010). *Software Engineering* (10th ed.). New York: Addison-Wesley Professional.

Spijkerman, W. (2010). *Tool Support for Change Impact Analysis in Requirement Models*. University of Twente.

Sun, X., Li, B., Tao, C., Wen, W., & Zhang, S. (2010). Change impact analysis based on a taxonomy of change types. In *International Computer Software and Applications Conference* (pp. 373–382). IEEE. https://doi.org/10.1109/COMPSAC.2010.45

Sünnetcioglu, A., Brandenburg, E., Rothenburg, U., & Stark, R. (2016). *ModelTracer: User-friendly Traceability for the Development of Mechatronic Products*. *Procedia Technology*, 26, (pp. 365–373). Elsevier. https://doi.org/10.1016/j.protcy.2016.08.047

Suzuki, N. (2002). A Structural Merging Algorithm for Xml documents. In *International Conference on WWW/Internet* (pp. 699–703). IADIS.

Tang, A., Jin, Y., & Han, J. (2007). A rationale-based architecture model for design traceability and reasoning. *Journal of Systems and Software*, *80*(6), 918–934. https://doi.org/10.1016/j.jss.2006.08.040

Thommazo, A. Di, Malimpensa, G., De Oliveira, T. R., Olivatto, G., & Fabbri, S. C. P. F. (2012). Requirements Traceability Matrix: Automatic Generation and Visualization. In *26th Brazilian Symposium on Software Engineering, SBES 2012* (pp. 101–110). Natal, Brazil: IEEE. https://doi.org/10.1109/SBES.2012.29

Tóth, G., Hegedűs, P., Beszédes, Á., Gyimóthy, T., & Jász, J. (2010). Comparison of different impact analysis methods and programmer's opinion: an empirical study. In *8th International Conference on the Principles and Practice of Programming in Java - PPPJ '10* (pp. 109–118). Vienna, Austria: ACM. https://doi.org/10.1145/1852761.1852777

Travis CI. (2018). Retrieved July 5, 2017, from https://travis-ci.org/

Trello. (2018). Retrieved October 2, 2018, from https://trello.com/

Trung, P. T., & Thang, H. Q. (2009). Building the reliability prediction model of component-based software architectures. *World Academy of Science, Engineering and Technology*, *35*(11), 911–918.

Tyree, J., & Akerman, A. (2005). Architecture decisions: Demystifying architecture. *IEEE Software*, *22*(2), 19–27. https://doi.org/10.1109/MS.2005.27

Vector Space Model. (2017). Retrieved July 3, 2017, from http://cogsys.imm.dtu.dk/thor/projects/multimedia/textmining/node5.html

Vrignat, P., Avila, M., Duculty, F., & Kratz, F. (2015). Failure Event Prediction Using Hidden Markov Model Approaches. *IEEE Transactions on Reliability*, *64*(3), 1038–1048. https://doi.org/10.1109/TR.2015.2423191

Wang, W., He, Y., Li, T., Zhu, J., & Liu, J. (2018). An Integrated Model for Information Retrieval Based Change Impact Analysis. *Scientific Programming*, *2018*(Article ID 5913634), 1–13. https://doi.org/10.1155/2018/5913634

Wang, Y., DeWitt, D. J., & Cai, J. Y. (2003). X-Diff: An effective change detection algorithm for XML documents. In *International Conference on Data Engineering* (pp. 519–530). IEEE. https://doi.org/10.1109/ICDE.2003.1260818

Wang, Y., Zhang, J., & Fu, Y. (2017). Rule-Based Change Impact Analysis Method in Software Development. In *2nd International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2017)* 74, (pp. 396–403). Atlantis Press.

Wijesinghe, D. B., Kamalabalan, K., Uruththirakodeeswaran, T., Thiyagalingam, G., Perera, I., & Meedeniya, D. (2014). Establishing traceability links among software artefacts. In *14th International Conference on Advances in ICT for Emerging Regions (ICTer)* (pp. 55–62). IEEE. https://doi.org/10.1109/ICTER.2014.7083879

Winkler, S. (2008). On Usability in Requirements Trace Visualizations. In *Requirements Engineering Visualization* (pp. 56–60). Catalunya, Spain: IEEE. https://doi.org/10.1109/REV.2008.4

Wong, S., Cai, Y., & Dalton, M. (2011). *Change Impact Analysis with Stochastic Dependencies*. *Technical Report*. PA, USA.

XMLUnit. (2018). Retrieved October 20, 2018, from https://www.xmlunit.org/

YAKINDU Traceability. (2019). Retrieved January 25, 2019, from https://www.itemis.com/en/yakindu/traceability/

Zeugmann, T., Poupart, P., Kennedy, J., Jin, X., Han, J., Saitta, L., … Fürnkranz, J. (2011). Precision and Recall. In *Encyclopedia of Machine Learning* (pp. 781–781). Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-30164-8_652

Zhang, S., Gu, Z., Lin, Y., & Zhao, J. (2008). Change impact analysis for AspectJ programs. In *IEEE International Conference on Software Maintenance* (pp. 87–96). Beijing, China: IEEE. https://doi.org/10.1109/ICSM.2008.4658057

Zhang, Y., Wan, C., & Jin, B. (2016). An empirical study on recovering requirement-to-code links. In *17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)* (pp. 121–126). Shanghai, China: IEEE. https://doi.org/10.1109/SNPD.2016.7515889

Zimmermann, T., Zeller, A., Weissgerber, P., & Diehl, S. (2005). Mining version histories to guide software changes. *IEEE Transactions on Software Engineering*, *31*(6), 429–445. https://doi.org/10.1109/TSE.2005.72

Zoho Sprints. (2018). Retrieved October 2, 2018, from https://www.zoho.com/sprints/