References

[1] Mello, John P. Jr. (2017) Application Security Report 2017. Cybersecurity Ventures.

[2] L. Gazzola, D. Micucci, and L. Mariani, "Automatic Software Repair: A Survey," *IEEE Transactions on Software Engineering*, vol. 45, no. 1, pp. 34–67, 2019.

[3] Krasner, Herb. (2018, 9) The Cost of Poor Quality Software in the US: A 2018 Report.

[4] "IEEE Standard Glossary of Software Engineering Terminology," *IEEE Std 610.12-1990*, pp. 1–84, 1990. doi: 10.1109/IEEESTD.1990.101064

[5] "IEEE Standard Classification for Software Anomalies," *IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993)*, pp. 1–23, 2010. doi: 10.1109/IEEESTD.2010.5399061

[6] M. Beller, R. Bholanath, S. McIntosh, and A. Zaidman, "Analyzing the State of Static Analysis: A Large-Scale Evaluation in Open Source Software," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, 2016, pp. 470–481.

[7] K. El Emam and I. Wieczorek, "The repeatability of code defect classifications," in *Proceedings Ninth International Symposium on Software Reliability Engineering (Cat. No.98TB100257)*, 1998. doi: 10.1109/ISSRE.1998.730897 pp. 322–333.

[8] J. Jones, "Abstract Syntax Tree Implementation Idioms," *Pattern Languages of Program Design*, 2003, proceedings of the 10th Conference on Pattern Languages of Programs (PLoP2003) http://hillside.net/plop/plop2003/papers.html. [Online]. Available: http://hillside.net/plop/plop2003/Papers/Jones-ImplementingASTs.pdf

[9] O. Kononenko, O. Baysal, and M. W. Godfrey, “Code review quality: How developers see it,” in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, 2016. doi: 10.1145/2884781.2884840 pp. 1028–1038.

[10] T. Baum, H. Leßmann, and K. Schneider, “The choice of code review process: A survey on the state of the practice,” in *Product-Focused Software Process Improvement*, M. Felderer, D. Méndez Fernández, B. Turhan, M. Kalinowski, F. Sarro, and D. Winkler, Eds. Cham: Springer International Publishing, 2017. ISBN 978-3-319-69926-4 pp. 111–127.

[11] S. C. Johnson, “Lint, a C Program Checker,” in *COMP. SCI. TECH. REP*, 1978, pp. 78–1273.

[12] B. Chess and J. West, *Secure Programming with Static Analysis*, 1st ed. Addison-Wesley Professional, 2007. ISBN 9780321424778

[13] S. Chacon and B. Straub, *Pro Git*, 2nd ed. USA: Apress, 2014. ISBN 1484200772

[14] S. D. Galup, R. Dattero, J. J. Quan, and S. Conger, “An Overview of IT Service Management,” *Commun. ACM*, vol. 52, no. 5, p. 124–127, May 2009. doi: 10.1145/1506409.1506439. [Online]. Available: https://doi.org/10.1145/1506409.1506439

[15] V. A. Danciu, A. Hanemann, M. Sailer, and H.-G. Hegering, *IT Service Management: Getting the View"*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 109–130. ISBN 978-3-540-34129-1. [Online]. Available: https://doi.org/10.1007/3-540-34129-3_7

[16] M. Brenner, M. Garschhammer, and H.-G. Hegering, *When Infrastructure Management Just Won’t Do: The Trend Towards Organizational IT Service Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 131–146. ISBN 978-3-540-34129-1. [Online]. Available: https://doi.org/10.1007/3-540-34129-3_8

[17] C. Vassallo, S. Panichella, F. Palomba, S. Proksch, H. Gall, and A. Zaidman, “How developers engage with static analysis tools in different contexts,” *Empirical Software Engineering*, vol. 25, 11 2019. doi: 10.1007/s10664-019-09750-5

[18] K. F. Tómasdóttir, M. Aniche, and A. Van Deursen, "The Adoption of JavaScript Linters in Practice: A Case Study on ESLint," *IEEE Transactions on Software Engineering*, vol. 46, no. 8, pp. 863–891, 2020. doi: 10.1109/TSE.2018.2871058

[19] C. Vassallo, S. Panichella, F. Palomba, S. Proksch, A. Zaidman, and H. C. Gall, "Context is king: The developer perspective on the usage of static analysis tools," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2018, pp. 38–49.

[20] S. Kim and M. D. Ernst, "Which Warnings Should I Fix First?" 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*, ser. ESEC-FSE '07. New York, NY, USA: Association for Computing Machinery, 2007. doi: 10.1145/1287624.1287633. ISBN 9781595938114 p. 45–54. [Online]. Available: https://doi.org/10.1145/1287624.1287633

[21] G. Digkas, M. Lungu, P. Avgeriou, A. Chatzigeorgiou, and A. Ampatzoglou, "How do developers fix issues and pay back technical debt in the Apache ecosystem?" in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2018, pp. 153–163.

[22] D. Marcilio, R. Bonifácio, E. Monteiro, E. Canedo, W. Luz, and G. Pinto, "Are Static Analysis Violations Really Fixed? A Closer Look at Realistic Usage of SonarQube," in *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, 2019, pp. 209–219.

[23] N. Imtiaz, B. Murphy, and L. Williams, "How Do Developers Act on Static Analysis Alerts? An Empirical Study of Coverity Usage," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 2019. doi: 10.1109/ISSRE.2019.00040 pp. 323–333.

[24] N. Fenton and N. Ohlsson, "Ohlsson, N.: Quantitative analysis of faults and failures in a complex software system. IEEE Trans. Softw. Eng. 26(8), 797-814," *Software Engineering, IEEE Transactions on*, vol. 26, pp. 797 – 814, 09 2000. doi: 10.1109/32.879815

[25] T. Galinac Grbac, P. Runeson, and D. Huljenic, "A Second Replicated Quantitative Analysis of Fault Distributions in Complex Software

Systems," *IEEE Transactions on Software Engineering*, vol. 39, pp. 462 – 476, 07 2012. doi: 10.1109/TSE.2012.46

[26] N. Walkinshaw and L. Minku, "Are 20% of Files Responsible for 80% of Defects?" in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '18. New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3239235.3239244. ISBN 9781450358231. [Online]. Available: https://doi.org/10.1145/3239235.3239244

[27] F. Zampetti, S. Scalabrino, R. Oliveto, G. Canfora, and M. Di Penta, "How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 2017. doi: 10.1109/MSR.2017.2 pp. 334–344.

[28] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why Don't Software Developers Use Static Analysis Tools to Find Bugs?" in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. IEEE Press, 2013. ISBN 9781467330763 p. 672–681.

[29] N. Imtiaz, A. Rahman, E. Farhana, and L. Williams, "Challenges with Responding to Static Analysis Tool Alerts," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 2019. doi: 10.1109/MSR.2019.00049 pp. 245–249.

[30] U. Yüksel and H. Sözer, "Automated Classification of Static Code Analysis Alerts: A Case Study," in *2013 IEEE International Conference on Software Maintenance*, 2013. doi: 10.1109/ICSM.2013.89 pp. 532–535.

[31] M. Monperrus, "Automatic Software Repair: A Bibliography," *ACM Comput. Surv.*, vol. 51, no. 1, Jan. 2018. doi: 10.1145/3105906. [Online]. Available: https://doi.org/10.1145/3105906

[32] T. Durieux, F. Madeiral, M. Martinez, and R. Abreu, "Empirical Review of Java Program Repair Tools: A Large-Scale Experiment on 2,141 Bugs and 23,551 Repair Attempts," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2019.

New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3338906.3338911. ISBN 9781450355728 p. 302–313. [Online]. Available: https://doi.org/10.1145/3338906.3338911

[33] C. L. Goues, M. Pradel, and A. Roychoudhury, "Automated Program Repair," *Commun. ACM*, vol. 62, no. 12, p. 56–65, Nov. 2019. doi: 10.1145/3318162. [Online]. Available: https://doi.org/10.1145/3318162

[34] M. Martinez, T. Durieux, R. Sommerard, J. Xuan, and M. Monperrus, "Automatic Repair of Real Bugs in Java: A Large-Scale Experiment on the Defects4J Dataset," *Empirical Software Engineering*, vol. 22, 08 2017. doi: 10.1007/s10664-016-9470-4

[35] K. Liu, D. Kim, T. F. Bissyandé, S. Yoo, and Y. Le Traon, "Mining Fix Patterns for FindBugs Violations," *IEEE Transactions on Software Engineering*, vol. 47, no. 1, pp. 165–188, 2021. doi: 10.1109/TSE.2018.2884955

[36] D. Marcilio, C. A. Furia, R. Bonifácio, and G. Pinto, "Automatically Generating Fix Suggestions in Response to Static Code Analysis Warnings," in *2019 19th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2019. doi: 10.1109/SCAM.2019.00013 pp. 34–44.

[37] E. Freeman, *DevOps For Dummies*, 1st ed. Newark: Wiley, 2019. ISBN 9781119552222

[38] N. Ayewah and W. Pugh, "The Google FindBugs Fixit," in *Proceedings of the 19th International Symposium on Software Testing and Analysis*, ser. ISSTA '10. New York, NY, USA: Association for Computing Machinery, 2010. doi: 10.1145/1831708.1831738. ISBN 9781605588230 p. 241–252. [Online]. Available: https://doi.org/10.1145/1831708.1831738

[39] J. Bader, A. Scott, M. Pradel, and S. Chandra, "Getafix: Learning to Fix Bugs Automatically," *Proc. ACM Program. Lang.*, vol. 3, no. OOPSLA, Oct. 2019. doi: 10.1145/3360585. [Online]. Available: https://doi.org/10.1145/3360585

Appendix A

Survey

This appendix contains the questionnaire created and used for the survey conducted in this study.

Section 1 – Base information about you and your current project

1. Experience as developer?
    - <1 year / 1–2 years / 2–5 years / 5–10 years / >10 years
2. Name of the main project / application you're working on? (Optional-free text)
3. The maturity of the project you're working on? (#years)
    - <1 year / 1–2 years / 2–5 years / 5–10 years / >10 years
4. The size of the team you're working in?
    - 1–3 developers / 4–6 developers / >6 developers
5. Programming languages used in the project? (Multiple choice list)
6. Level of CI/CD in the project? (To what degree the project has adopted CI/CD principles)
    - Manual builds and manual deployments
    - Automated builds, tests, and manual deployments
    - Automated builds, tests, and automated deployments to dev/test
    - Fully automated continuous deployments