REFERENSI

[1] Kementerian Pariwisata dan Ekonomi Kreatif, "Outlook Ekraf 2020-2021," Jakarta, Indonesia, 2020.

[2] Kementerian Perindustrian (Kemenperin), "*Pemerintah Pacu Perkembangan Industri Software,"* 2018. [Online]. Available: https://kemenperin.go.id/artikel/18661/Pemerintah-Pacu-Perkembangan-Industri-Software.

[3] P. Ammann and J. Offutt, *Introduction to Software Testing*. 2008.

[4] M. Usman, E. Mendes, and J. Börstler, "The Economic Impacts of Inadequate Infrastructure for Software Testing," 2015.

[5] N.G. Leveson and C.S. Turner, "An investigation of the Therac-25 accidents," *Computer (Long. Beach. Calif).*, vol. 26, no. 7, pp. 18–41, 1993, doi: 10.1109/MC.1993.274940.

[6] R. . Charette, "Why software fails," *IEEE Spectr. J.*, vol. 42, no. 9, pp. 42–49, 2002.

[7] E. Bounimova, P. Godefroid, and D. Molnar, "Billions and billions of constraints: Whitebox fuzz testing in production," *Proc. - Int. Conf. Softw. Eng.*, pp. 122–131, 2013, doi: 10.1109/ICSE.2013.6606558.

[8] L. Williams, G. Kudrjavets, and N. Nagappan, "On the effectiveness of unit test automation at microsoft," in *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, 2009, pp. 81–89, doi: 10.1109/ISSRE.2009.32.

[9] A. Tosun, M. Ahmed, B. Turhan, and N. Juristo, "On the effectiveness of unit tests in test-driven development," in *ACM International Conference Proceeding Series*, 2018, pp. 113–122, doi: 10.1145/3202710.3203153.

[10] E. Daka and G. Fraser, "A survey on unit testing practices and problems," in

Proceedings - International Symposium on Software Reliability Engineering, ISSRE, 2014, pp. 201–211, doi: 10.1109/ISSRE.2014.11.

[11] M. Beller, G. Gousios, A. Panichella, S. Proksch, S. Amann, and A. Zaidman, "Developer Testing in the IDE: Patterns, Beliefs, and Behavior," *IEEE Trans. Softw. Eng.*, vol. 45, no. 3, pp. 261–284, 2019, doi: 10.1109/TSE.2017.2776152.

[12] P. Louridas, "JUnit: Unit testing and coding in tandem," *IEEE Softw.*, vol. 22, no. 4, pp. 12–15, Jul. 2005, doi: 10.1109/MS.2005.100.

[13] "JUnit in Action | Guide books." [Online]. Available: https://dl.acm.org/doi/book/10.5555/961868. [Accessed: 20-Feb-2020].

[14] "NUnit." [Online]. Available: https://www.nunit.org/. [Accessed: 20-Feb-2020].

[15] N. Setiani, R. Ferdiana, P. I. Santosa, and R. Hartanto, "Literature review on test case generation approach," in *ACM International Conference Proceeding Series*, 2019, doi: 10.1145/3305160.3305186.

[16] G. Fraser and A. Arcuri, "1600 faults in 100 projects: automatically finding faults while achieving high coverage with EvoSuite," *Empir. Softw. Eng.*, vol. 20, no. 3, pp. 611–639, Jun. 2015, doi: 10.1007/s10664-013-9288-2.

[17] C. Pacheco and M. D. Ernst, "Randoop: Feedback-directed random testing for Java," *Proc. Conf. Object-Oriented Program. Syst. Lang. Appl. OOPSLA*, pp. 815–816, 2007, doi: 10.1145/1297846.1297902.

[18] S. Vogl, S. Schweikl, G. Fraser, A. Arcuri, J. Campos, and A. Panichella, "Evosuite at the SBST 2021 Tool Competition," *Proc. - 2021 IEEE/ACM 14th Int. Work. Search-Based Softw. Testing, SBST 2021*, pp. 28–29, 2021, doi: 10.1109/SBST52555.2021.00012.

[19] A. Arcuri, "An experience report on applying software testing academic results in industry: we need usable automated test generation," *Empir. Softw.*

Eng., vol. 23, no. 4, pp. 1959–1981, Aug. 2018, doi: 10.1007/s10664-017-9570-9.

[20] G. Fraser, M. Staats, P. McMinn, A. Arcuri, and F. Padberg, "Does automated unit test generation really help software testers? A controlled empirical study," in *ACM Transactions on Software Engineering and Methodology*, 2015, vol. 24, no. 4, pp. 1–49, doi: 10.1145/2699688.

[21] D. Honfi and Z. Micskei, "Classifying generated white-box tests: an exploratory study," *Software Quality Journal*, 2019. [Online]. Available: https://zenodo.org/record/2596044#.XkI4fhMzbUp. [Accessed: 11-Feb-2020].

[22] G. Fraser, M. Staats, P. McMinn, A. Arcuri, and F. Padberg, "Does automated white-box test generation really help software testers?," *2013 Int. Symp. Softw. Test. Anal. ISSTA 2013 - Proc.*, no. July 2013, pp. 291–301, 2013, doi: 10.1145/2483760.2483774.

[23] B. Souza and P. Machado, "A Large Scale Study On the Effectiveness of Manual and Automatic Unit Test Generation," in *SBES '20: Proceedings of the 34th Brazilian Symposium on Software Engineering*, 2020, pp. 253–262, doi: https://doi.org/10.1145/3422392.3422407.

[24] S. Scalabrino, G. Bavota, C. Vendome, M. Linares-Vasquez, D. Poshyvanyk, and R. Oliveto, "Automatically Assessing Code Understandability," *IEEE Trans. Softw. Eng.*, 2019, doi: 10.1109/TSE.2019.2901468.

[25] H. Khanh, V. Tran, and N. Bin Ali, "Test-Case Quality – Understanding Practitioners ' Perspectives," pp. 37–52, 2019.

[26] G. Grano, S. Scalabrino, H. C. Gall, and R. Oliveto, "An empirical investigation on the readability of manual and generated test cases," in *Proceedings - International Conference on Software Engineering*, 2018, pp.

348–351, doi: 10.1145/3196321.3196363.

[27] E. Daka, J. Campos, G. Fraser, J. Dorn, and W. Weimer, "Modeling readability to improve unit tests," in *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*, 2015, pp. 107–118, doi: 10.1145/2786805.2786838.

[28] A. Tosun, O. Dieste, D. Fucci, S. Vegas, and B. Turhan, "An industry experiment on the effects of test-driven development on external quality and productivity," *Empir. Softw. Eng.*, vol. 22, pp. 2763–280, 2017.

[29] N. C. Borle, M. Feghhi, E. Stroulia, R. Greiner, and A. Hindle, "Analyzing the effects of test driven development in GitHub," in *ICSE '18: Proceedings of the 40th International Conference on Software Engineering*, 2018, p. 1062, doi: https://doi.org/10.1145/3180155.3182535.

[30] A. Panichella, F. M. Kifetew, and P. Tonella, "Automated Test Case Generation as a Many-Objective Optimisation Problem with Dynamic Selection of the Targets," *IEEE Trans. Softw. Eng.*, vol. 44, no. 2, pp. 122–158, Feb. 2018, doi: 10.1109/TSE.2017.2663435.

[31] W. E. Howden, "Symbolic Testing and the DISSECT Symbolic Evaluation Systerm," *IEEE Trans. Softw. Eng.*, vol. SE-3, no. 4, pp. 266–278, 1977, doi: 10.1109/TSE.1977.231144.

[32] M. Papadakis, M. Kintis, J. Zhang, Y. Jia, Y. Le Traon, and M. Harman, "Mutation Testing Advances: An Analysis and Survey," *Adv. Comput.*, vol. 112, pp. 275–378, Jan. 2019, doi: 10.1016/bs.adcom.2018.03.015.

[33] M. P. Prado and A. M. R. Vincenzi, "Towards cognitive support for unit testing: A qualitative study with practitioners," *J. Syst. Softw.*, vol. 141, pp. 66–84, 2018, doi: 10.1016/j.jss.2018.03.052.

[34] B. Marculescu, R. Feldt, R. Torkar, and S. Poulding, "Transferring

interactive search-based software testing to industry," *J. Syst. Softw.*, vol. 142, no. April, pp. 156–170, 2018, doi: 10.1016/j.jss.2018.04.061.

[35] D. Amalfitano, A. R. Fasolino, P. Tramontana, B. D. Ta, and A. M. Memon, "MobiGUITAR: Automated Model-Based Testing of Mobile Apps," *IEEE Softw.*, vol. 32, no. 5, pp. 53–59, Sep. 2015, doi: 10.1109/MS.2014.55.

[36] S. Panichella, A. Gambi, F. Zampetti, and V. Riccio, "SBST Tool Competition 2021," *Proc. - 2021 IEEE/ACM 14th Int. Work. Search-Based Softw. Testing, SBST 2021*, pp. 20–27, 2021, doi: 10.1109/SBST52555.2021.00011.

[37] G. Fraser and C. Science, "A Large Scale Evaluation of Automated Unit Test Generation Using EvoSuite," vol. V, no. 212.

[38] N. Tillmann and J. De Halleux, "Pex-white box test generation for .NET," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4966 LNCS, pp. 134–153, 2008, doi: 10.1007/978-3-540-79124-9_10.

[39] L. Ma, C. Artho, C. Zhang, H. Sato, J. Gmeiner, and R. Ramler, "GRT: An automated test generator using orchestrated program analysis," in *Proceedings - 2015 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015*, 2016, pp. 842–847, doi: 10.1109/ASE.2015.102.

[40] C. Csallner and Y. Smaragdakis, "JCrasher: An automatic robustness tester for Java," *Softw. - Pract. Exp.*, vol. 34, no. 11, pp. 1025–1050, 2004, doi: 10.1002/spe.602.

[41] W. Miller and D. L. Spooner, "Automatic Generation of Floating-Point Test Data," *IEEE Trans. Softw. Eng.*, vol. SE-2, no. 3, pp. 223–226, 1976.

[42] B. Korel, "Dynamic method for software test data generation," *Softw. Testing, Verif. Reliab.*, vol. 2, no. 4, pp. 203–213, Dec. 1992, doi:

10.1002/stvr.4370020405.

[43] P. McMinn, "Search-Based Software Testing: Past, Present and Future," in *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, 2011.

[44] R. J. Lipton and F. G. Sayward, "Hints on test data selection: Help for the practicing programmer," *Computer (Long. Beach. Calif).*, vol. 11, no. 4, pp. 34–41, 1978, doi: 10.1109/C-M.1978.218136.

[45] R. G. Hamlet, "Testing Programs with the Aid of a Compiler," *IEEE Trans. Softw. Eng.*, vol. SE-3, no. 4, pp. 279–290, 1977, doi: 10.1109/TSE.1977.231145.

[46] H. Zhu, P. A. V. Hall, and J. H. R. May, "Software unit test coverage and adequacy," *ACM Comput. Surv.*, vol. 29, no. 4, pp. 366–427, 1997, doi: 10.1145/267580.267590.

[47] R. Just, D. Jalali, L. Inozemtseva, M. D. Ernst, R. Holmes, and G. Fraser, "Are Mutants a Valid Substitute for Real Faults in Software Testing?," doi: 10.1145/2635868.2635929.

[48] K. Pavnett Singh, D. Lo, L. Julia, and N. Nagappan, "Code Coverage and Postrelease Defects: A Large-Scale Study on Open Source Projects," *IEEE Trans. Reliab.*, vol. 66, no. 4, pp. 1213–1228.

[49] N. Gupta, A. Sharma, and M. K. Pachariya, "An Insight into Test Case Optimization: Ideas and Trends with Future Perspectives," *IEEE Access*, vol. 7, pp. 22310–22327, 2019, doi: 10.1109/ACCESS.2019.2899471.

[50] D. Di Nardo, N. Alshahwan, L. Briand, and Y. Labiche, "Coverage-based regression test case selection, minimization and prioritization: A case study on an industrial system," in *Software Testing Verification and Reliability*, 2015, vol. 25, no. 4, pp. 371–396, doi: 10.1002/stvr.1572.

[51] B. Miranda and A. Bertolino, "Scope-aided test prioritization, selection and

minimization for software reuse," *J. Syst. Softw.*, vol. 131, pp. 528–549, 2017, doi: 10.1016/j.jss.2016.06.058.

[52] A. Zeller and R. Hildebrandt, "Simplifying and isolating failure-inducing input," *IEEE Trans. Softw. Eng.*, vol. 28, no. 2, pp. 183–200, 2002, doi: 10.1109/32.988498.

[53] E. Daka, J. M. Rojas, and G. Fraser, "Generating unit tests with descriptive names or: Would you name your children thing1 and thing2?," in *ISSTA 2017 - Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2017, pp. 57–67, doi: 10.1145/3092703.3092727.

[54] S. Panichella, A. Panichella, M. Beller, A. Zaidman, and H. C. Gall, "The impact of test case summaries on bug fixing performance: An empirical investigation," in *Proceedings - International Conference on Software Engineering*, 2016, vol. 14-22-May-2016, pp. 547–558, doi: 10.1145/2884781.2884847.

[55] S. Zhang, "Practical semantic test simplification," in *Proceedings - International Conference on Software Engineering*, 2013, pp. 1173–1176, doi: 10.1109/ICSE.2013.6606671.

[56] J. Brown, "I know what you did last summer," *Biochem. (Lond).*, vol. 39, no. 6, pp. 46–48, 2017, doi: 10.1042/bio03906046.

[57] C. Chen, R. Alfayez, K. Srisopha, L. Shi, and B. Boehm, "Evaluating Human-Assessed Software Maintainability Metrics," *Softw. Eng. Methodol. Emerg. Domains*, vol. 675, no. January 2016, p. V, 2016, doi: 10.1007/978-981-10-3482-4.

[58] S. Misra and I. Akman, "Comparative study of cognitive complexity measures," in *2008 23rd International Symposium on Computer and Information Sciences, ISCIS 2008*, 2008, doi: 10.1109/ISCIS.2008.4717939.

[59] S. Scalabrino, M. Linares-Vásquez, R. Oliveto, and D. Poshyvanyk, "A comprehensive model for code readability," *J. Softw. Evol. Process*, vol. 30, no. 6, p. e1958, Jun. 2018, doi: 10.1002/smr.1958.

[60] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Trans. Softw. Eng.*, vol. 28, no. 1, pp. 4–17, 2002, doi: 10.1109/32.979986.

[61] N. Kasto and J. Whalley, "Measuring the difficulty of code comprehension tasks using software metrics," 2013.

[62] K. Shima, Y. Takemura, and K. Matsumoto, "An approach to experimental evaluation of software understandability," in *ISESE 2002 - Proceedings, 2002 International Symposium on Empirical Software Engineering*, 2002, pp. 48–55, doi: 10.1109/ISESE.2002.1166925.

[63] A. Trockman, K. Cates, M. Mozina, T. Nguyen, C. Kästner, and B. Vasilescu, "'automatically assessing code understandability' reanalyzed: Combined metrics matter," *Proc. - Int. Conf. Softw. Eng.*, pp. 314–318, 2018, doi: 10.1145/3196398.3196441.