Empirical studies on the impact of test-first programming

Lech Madeyski
Wroclaw University of Technology
Wyb.Wyspianskiego 27, 50370 Wroclaw, POLAND
WWW: http://madeyski.e-informatyka.pl
E-mail: Lech.Madeyski@pwr.wroc.pl

February 21, 2009

Abstract

This paper presents productivity and quality effects of test-first programming technique in a tabular form and, as a result, provides a summary of many empirical studies conducted so far. The table has been moved here from one of my research papers according to the anonymous reviewer's suggestion.

1 Empirical studies on the impact of test-first (TF) programming

Both productivity and quality effects of TF technique are presented in Table 1 which provides a summary of many empirical studies conducted so far.

The table reports, from left to right, the references for each study, the environment (e.g. academic or industrial) in which an empirical study was conducted, the number of subjects involved in each empirical study, and statistical significance of the effects of TF practice (if available) along with textual descriptions of the findings.

Table 1: Empirical studies on the impact of test-first (TF) programming[0]

Studies	Environment, Subjects	Details of TF effects
Ynchausti [26]	Industrial, 5	• 38-267% increase in the quality test pass rate percentage
Müller&Hagner [20]	Academic, 19	• does not accelerate the implementation ✓ ($p = .03$) lower reliability in acceptance testing ($reliability = \frac{passed\ assertions}{all\ assertions}$) • slightly lower code coverage • does not aid the developer in a proper usage of existing code • seems to support better program understanding
Pancur [22]	Academic, 38	• small difference in external quality (external tests passed) • slightly lower code coverage
Abrahamsson et al. [1]	Academic/ Industrial, 4[1]	• little or no added value to a team perceived by developers • team used 0%(iteration 5)–30%(iteration 1) of effort for TF
Williams et al. [25] Maximilien& Williams [19]	Industrial, 9	• reduced defect rate by 40%[25]–50%[19] • minimal [19] or no difference [25] in $\frac{LOC}{person-month}$
George&Williams [8, 9]	Industrial, 24	• 16% longer development[2] • 18% more functional tests passed[2]
Geras et al. [10]	Academic, 14	• little or no difference in developer productivity
Erdogmus et al. [6]	Academic, 24	✓ ($p = .09$) on average 52% more tests • on average 28% more delivered user stories (USs)[3] • on average 2% less assertions passed from the acceptance tests[3]
Melnik&Maurer [11]	Academic, 240	• 73% of students perceived TF improves quality
Madeyski[16, 17]	Academic, 188	✓ ($p = .028$) significantly less acceptance tests passed ✓ ($p = .013$) significantly less acceptance tests passed[2] • package dependencies were not significantly affected
Flohr&Schneider [7]	Academic, 18	• 21% decrease in development time[2] • small difference in code coverage[2] • no difference in number of assertions written[2]
Canfora et al. [3]	Industrial, 28	✓ ($p < .05$) more time per assertion/overall time/time in average • no evidence of more assertions or more assertions per method
Bhat&Nagappan [2]	Industrial, 6(A) 5–8(B)	• 15%(project B)–35%(project A) longer development time • decreased $defects/KLOC$ by 62%(project A)–76% (project B)
Damm&Lundberg [4, 5]	Industrial, 100	• 5–30% decrease in fault-slip-through rate[4] • 60% decrease in avoidable fault costs[4] • total project cost became 5-6% less[4] • ratio of faults decreased from 60–70% to 0–20%[4] • cost savings in maintenance (up to 25% of the development cost)
Sanchez et al. [23]	Industrial, 9–17	• it took on average 15% or more[5] of overall time to write • unit tests reduced internal defect rate
Siniaalto& Abrahamsson [24]	Academic/ Industrial[6], 4,5,4	• slightly less coupled code (CBO metric) but results are dispersed • high lack of cohesion (LCOM metric) • WMC, DIT, NOC and RFC did not reveal significant differences

Studies	Environment, Subjects	Details of TF effects
		• higher method, statement and branch coverage levels
Madeyski [18]	Academic Industrial[7], 1	• higher ratio of active to passive development time[8] • increased LOC/h[9] • increased $number\ of\ user\ stories/h$[9] • increased $number\ of\ acceptance\ tests/h$[9]
Gupta&Jalote [12]	Academic, 22	✓ ($p = .001$) improves external code quality[10] (affected by the actual testing efforts) ✓ ($p = .02$) reduces overall development efforts[10] • improves developers productivity
Nagappan et al. [21][11]	Industrial, 9,6,5–8,7	• decreased defects rate by 40%–90% • 15%-35% longer development time
Janzen&Saiedian [15]	Industrial, 1,2,2,5/ Academic, 3,7	• possible tendency to write smaller, simpler classes& methods[12] • tendency to write simpler classes&sometimes simpler methods[12] • coupling analysis does not reveal clear answers • does not improve cohesion
Huang&Holcombe [14]	Academic,39	• does not influence external clients' assessment of quality • more effort on testing ($p < .1$) ◦ 70% higher productivity but the improvement is not statistically significant

[0] Abbreviations: A-Academic, I-Industrial, XXX-number of subjects, (e.g. "Academic,12" means empirical study in academic environment with 12 subjects), ✓ means statistically significant result (e.g. ✓$p < .05$)

[1] Three students with industrial experience and one industrial developer

[2] TF pairs vs. TL pairs

[3] only USs that passed at least 50% of the assert statements from the acceptance test suite were considered

[4] combined effect of introducing component-level test automation, as well as TF

[5] calculated based on questionnaires

[6] undergraduates but real projects

[7] an experienced programmer, with recent industrial experience, classified as E4 according to Höst et al. [13], developed a web-based system for academic institution

[8] the active time may be described as typing and producing code, while the passive time is spent on reading the source code, looking for a bug etc.

[9] not only TF, but also experience and knowledge of the application domain gained during the course of the project seem to drive productivity

[10] in one of the two programs

[11] builds up on the prior empirical work [2, 25, 19]

[12] in the case of some studies differences were statistically significant

References

[1] P. Abrahamsson, A. Hanhineva, and J. Jäälinoja. Improving business agility through technical solutions: A case study on test-driven development in mobile software development. In R. Baskerville, L. Mathiassen, J. Pries-Heje, and J. I. DeGross, editors, *Proceedings of the IFIP TC8 WG 8.6 International Working Conference on Business Agility and Information Technology Diffusion*, volume 180 of *IFIP International Federation for Information Processing*, pages 1–17. Springer, 2005.

[2] T. Bhat and N. Nagappan. Evaluating the efficacy of test-driven development: industrial case studies. In *ISESE '06: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, pages 356–363, New York, NY, USA, 2006. ACM Press.

[3] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. A. Visaggio. Evaluating advantages of test driven development: a controlled experiment with professionals. In *ISESE '06: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, pages 364–371, New York, NY, USA, 2006. ACM Press.

[4] L.-O. Damm and L. Lundberg. Results from introducing component-level test automation and Test-Driven Development. *Journal of Systems and Software*, 79(7):1001–1014, 2006.

[5] L.-O. Damm and L. Lundberg. Quality impact of introducing component-level test automation and test-driven development. In P. Abrahamsson, N. Baddoo, T. Margaria, and R. Messnarz, editors, *Software Process Improvement*, volume 4764 of *Lecture Notes in Computer Science*, pages 187–199. Springer, 2007.

[6] H. Erdogmus, M. Morisio, and M. Torchiano. On the Effectiveness of the Test-First Approach to Programming. *IEEE Transactions on Software Engineering*, 31(3):226–237, 2005.

[7] T. Flohr and T. Schneider. Lessons Learned from an XP Experiment with Students: Test-First Need More Teachings. In J. Münch and M. Vierimaa, editors, *Product Focused Software Process Improvement*, volume 4034 of *Lecture Notes in Computer Science*, pages 305–318, Berlin, Heidelberg, 2006. Springer.

[8] B. George and L. Williams. An initial investigation of test driven development in industry. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 1135–1139, New York, NY, USA, 2003. ACM.

[9] B. George and L. A. Williams. A structured experiment of test-driven development. *Information and Software Technology*, 46(5):337–342, 2004.

[10] A. Geras, M. R. Smith, and J. Miller. A Prototype Empirical Evaluation of Test Driven Development. In *IEEE METRICS'2004: Proceedings of the 10th IEEE International Software Metrics Symposium*, pages 405–416. IEEE Computer Society, 2004.

[11] Grigori Melnik and Frank Maurer. A cross-program investigation of students' perceptions of agile methods. In *ICSE '05: Proceedings of the 27th International Conference on Software Engineering*, pages 481–488, 2005.

[12] A. Gupta and P. Jalote. An experimental evaluation of the effectiveness and efficiency of the test driven development. In *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pages 285–294, Washington, DC, USA, 2007. IEEE Computer Society.

[13] M. Höst, , C. Wohlin, and T. Thelin. Experimental Context Classification: Incentives and Experience of Subjects. In *ICSE '05: Proceedings of the 27th International Conference on Software Engineering*, pages 470–478, New York, NY, USA, 2005. ACM Press.

[14] L. Huang and M. Holcombe. Empirical investigation towards the effectiveness of Test First programming. *Information and Software Technology*, 51(1):182–194, 2009.

[15] D. Janzen and H. Saiedian. Does Test-Driven Development Really Improve Software Design Quality? *IEEE Software*, 25(2):77–84, March–April 2008.

[16] L. Madeyski. Preliminary Analysis of the Effects of Pair Programming and Test-Driven Development on the External Code Quality. In

K. Zieliński and T. Szmuc, editors, *Software Engineering: Evolution and Emerging Technologies*, volume 130 of *Frontiers in Artificial Intelligence and Applications*, pages 113–123. IOS Press, 2005.

[17] L. Madeyski. The Impact of Pair Programming and Test-Driven Development on Package Dependencies in Object-Oriented Design – An Experiment. In J. Münch and M. Vierimaa, editors, *Product Focused Software Process Improvement*, volume 4034 of *Lecture Notes in Computer Science*, pages 278–289, Berlin, Heidelberg, 2006. Springer. http://madeyski.e-informatyka.pl/download/Madeyski06.pdf.

[18] L. Madeyski and Ł. Szała. The Impact of Test-Driven Development on Software Development Productivity – An Empirical Study. In P. Abrahamsson, N. Baddoo, T. Margaria, and R. Messnarz, editors, *Software Process Improvement*, volume 4764 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2007. http://madeyski.e-informatyka.pl/download/Madeyski07d.pdf.

[19] E. M. Maximilien and L. A. Williams. Assessing Test-Driven Development at IBM. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 564–569. IEEE Computer Society, 2003.

[20] M. M. Müller and O. Hagner. Experiment about test-first programming. *IEE Procedings-Software*, 149(5):131–136, 2002.

[21] N. Nagappan, E. M. Maximilien, T. Bhat, and L. Williams. Realizing quality improvement through test driven development: results and experiences of four industrial teams. *Empirical Software Engineering*, 13(3), 2008.

[22] M. Pančur, M. Ciglarič, M. Trampuš, and T. Vidmar. Towards empirical evaluation of test-driven development in a university environment. In *EUROCON '03: Proceedings of the International Conference on Computer as a Tool*, pages 83–86, 2003.

[23] J. C. Sanchez, L. Williams, and E. M. Maximilien. On the sustained use of a test-driven development practice at ibm. In *AGILE '07: Proceedings of the 2007 Conference on Agile Software Development*, pages 5–14, Washington, DC, USA, 2007. IEEE Computer Society.

[24] M. Siniaalto and P. Abrahamsson. A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage. In *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pages 275–284. IEEE Computer Society, 2007.

[25] L. Williams, E. M. Maximilien, and M. Vouk. Test-Driven Development as a Defect-Reduction Practice. In *ISSRE '03: Proceedings of the 14th International Symposium on Software Reliability Engineering*, pages 34–48, Washington, DC, USA, 2003. IEEE Computer Society.

[26] R. A. Ynchausti. Integrating Unit Testing Into A Software Development Team's Process. In M. Marchesi and G. Succi, editors, *XP 2001: Proceedings of the 2nd International Conference on Extreme Programming and Flexible Processes in Software Engineering*, pages 84–87, Sardinia, Italy, 2001.