An Empirical Study on the Evolution of Test Smell

Dong Jae Kim
djaekim086@gmail.com
Concordia University
Montreal, Quebec

ABSTRACT

Test smell as analogous to code smell is a poor design choice in the implementation of test code. Recently, the concept of test smell has become the utmost interest of researchers and practitioners. Surveys show that developers' are aware of test smells and their potential consequences in the software system. However, there is limited empirical evidence for how developers address test smells during software evolution. Thus, in this paper, we study 2 research questions: (RQ1) How do test smells evolve? (RQ2) What is the motivation for removing test smells? Our result shows that Assertion Roulette, Conditional Test Logic and Unknown tests have a high rate of churns, the feature addition and improvement motivate refactoring, but test smell persists, implicating sub-optimal practice. In our study, we hope to fill the gap between academia and industry by providing evidence of sub-optimal practice in the way developers address test smells, and how it may be detrimental to the software.

ACM Reference Format:
Dong Jae Kim. 2020. An Empirical Study on the Evolution of Test Smell. In *Proceedings of ICSE '2020: the 42nd International Conference on Software Engineering (ICSE '2020).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Software testing an essential part of software development and plays a crucial role in software quality assurance [1]. Test cases are useful in finding bugs early in software development and is used to continuously validate the software quality for detecting regressions [4, 6, 10]. However, similar to the production code, there may also be quality issues in the test code. For example, prior studies found that the results of some test cases may be unreliable (e.g., flaky tests) due to bugs in test code [13].

Recently, researchers and practitioners have started to notice design problems in the test code [3, 7, 11, 14]. Bavota et al. [3] found that test smells are prevalent in software systems, and such test smells may hinder program comprehension and maintenance. Palomba et al. [8] found that some test smells may cause flaky tests that affect the quality of the test code. However, even though a recent survey found that developers are aware of test smells and their potential consequences [5], it is still not clear if developers address test smells during software development, and whether fixing the test smells would have an effect on the quality of the production code.

ICSE '2020, May 23–29, 2020, Seoul, South Korea

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

Table 1: An overview of our studied systems.

System	Version	LOC in Production	LOC in Test	Added Smell	Removed Smell
Kafka	2.0.0-2.1.1	94K - 117K	81K - 110K	313	146
Flink	1.4.1-1.6.1	260K - 332K	268K - 339K	580	363
Jclouds	1.2.0-1.4.0	172K	102K	125	37
Hive	2.3.4-2.3.5	207K - 888K	18K - 152K	562	211
Wicket	8.2.0-8.4.0	127K - 141K	50K - 74K	261	192
Accumulo	1.6.6-1.7.3	170K - 334K	29K - 84K	44	11
Hadoop	3.1.1-3.1.2	729K - 801K	665K - 728K	57	10
Cassandra	3.11.3-3.11.4	171K - 287K	22K - 123K	565	342

In this paper, we conduct a preliminary study on the maintenance of test smells. We hope to fill the gap between industry and academia by providing evidence for how developers address test smells and whether the sub-optimal practice of addressing test smells may be detrimental in software. We first quantify the different types of test smells that were added and removed during software evolution, and qualitatively study the removed test smells to gain insights on what motivates developers to address test smells. Our result shows that Assertion Roulette, Conditional Test Logic and Unknown tests have a high rate of churns, the feature addition and improvements motivate refactoring, but test smell persists, implicating sub-optimal practice. In the future study, we expand our qualitative analysis on what motivates developers to remove test smells and revisit a study on the relationship between test smells on software quality by studying how the test smell evolution may contribute to a better explanation for the post-release defect. The rest of the paper is organized as follows. Section 2 discusses experimental setup, Section 3 provides the preliminary result of our research questions, and Section 4 discusses future work and concludes the paper.

2 EXPERIMENTAL SETUP

Studied Systems. Table 1 shows an overview of our studied systems. We conduct our study on eight large-scale open-source Java systems. We choose these systems because they are large in scale, well maintained, and studied in prior test smell research [2, 5, 11]. These systems also cover various domains and are used in many commercial settings. Since studying test code is essential for our study, we also choose systems based on having greater than 1000 test cases. Finally, we choose release versions based on 2 criteria. (i) We only consider systems with significant reported issues based

Table 2: Five Most Commonly Added (+) and Removed Smells (-).

	Unknown Test	Assertion Roulette	Condition Test Logic	Duplicate Assert	Exception Catch
No. of (+)	275(11%)	178(7%)	201(8%)	158(6.3%)	144(5.7%)
	Assertion Roulette	Condition Test Logic	Magic Number Test	Unknown Test	Duplicate Assert
No. of (-)	115(8.8%)	109(8.3%)	78(5.9%)	47(3.6%)	44(3.3%)

Table 3: Qualitative Analysis: Motivations for Removing Test Smell

Motivation	Occurrences
Deprecates/updates mock object and changes setup fixture and test code.	(5)
Creates a helper method by extracting reusable method and adds additional assertion statements.	(2)
Developer writes "Refactoring" in commit message, but ends up deleting code base.	(4)
In feature addition, extract reusable method to improve testability and eases maintainbility.	(1)
Improve to a new driver test	(2)

on release documentation since we are also studying test smells evolution from maintenance activities. (ii) Since minor releases are closer to each other, we choose releases that are 6 months apart because we believed it is sufficient to capture a series of maintenance efforts and associated test smell evolutions.

Data preparation. We use the test smell detection tool developed in a prior study [9]. The tool uses static analysis to detect 19 different test smells and reported to have very high precision and recall (0.93 and 0.98, respectively). Among the studied test smells, we only report the top 5 most churned test smells, such as 'Assertion Roulette', 'Conditional Test Logic', 'Duplicate Assert', 'Unknown Test' and 'Exception Catch'. Due to space restrictions, we omit the rest of the studied test smells, but the full list and descriptions of test smells are discussed in prior work [9]. The final data includes all the test files labelled with commit hash, and types of test smell added and removed during software evolution.

3 RESEARCH QUESTIONS

RQ1: How do test smells evolve?

Approach We first extract all test files in all releases of studied systems. Since studying on a commit-level basis is computationally expensive, we leveraged the command "git follow" to extract commits that modified the test files. For the mined commits, we run the test smell detection tool, identifying and labelling the presence of the 19 types of test smells. We then use the result to calculate test smell addition or removal for all test files.

Preliminary results. In Table 1, we see that for all the studied systems, the test smell added is twice as much as smells removed, suggesting that developers typically introduce more test smells during software evolution. Table 2 shows the five most commonly added and removed test smells among the studied systems. We see that Unknown Test (11%), Assertion Roulette (7%), Conditional Test Logic (8%), Duplicate Assertion (6.3%) and Exception Catch (5.7%) are the most commonly added smells and Assertion Roulette (8.8%), Conditional Test Logic (8.3%), Magic Number Test (5.9%), Unknown Test (3.6%)and Duplicate Asserts (3.3%) are the most commonly removed smells. Previous works show that Assertion Roulette, Duplicate Assertion and Conditional Test Logic occurred the most in software compared to other smells [9]. Similarly, we see from Table 2 that the considered test smells also have the highest churn rates in evolution. One of our future works is to revisit the work by Spadini et al. [11] to study how the evolution (i.e., addition and removal) of test smells may affect software quality.

RQ2: What is the motivation for removing test smells?

Approach. We randomly sampled 50 commits for our manual analysis. We use the bug reports and commit messages to understand motivations for removal. To enhance our qualitative study, we also leveraged a tool called Refactoring Aware Commit Review [12] which is a diff visualization tool for showing the refactoring activities applied between two commits.

Preliminary results. Table 3 shows our studied motivation for smell removal. In particular, deprecating/updating the mock object was the most common reason. For example, the developers used @before to reduce the number of mock instantiation to ease deprecation. More interestingly, some of the discussed motivations also influenced test refactorings, which removes test smells. However, for extract method and extract class refactorings, we always saw the persistence of test smells in the extracted code, or the addition of new test smells when the extracted code adds new features, showing evidence of sub-optimal practice caused by unawareness of test smells. Some commit also contained "Refactor" in the commit messages, but there was, in fact, no known refactorings. For example, existing methods were replaced with a new method or deleted. The rest of the studied commits were general maintenance tasks related to feature addition and improvements. The type of test smells removed varied, and in the future, we plan on expanding our sample to generalize some interesting co-occurrences of removed test smells and motivations.

4 CONCLUSION

Test smell is a poor design choice in the implementation of test code [14]. Failure to maintain a high-quality test code increase the diffusion of hidden bugs and incur more cost for software systems [4, 6, 10]. Our result shows the common test smells added and removed in software evolution, show that feature addition motivates test refactoring, but test smell persists, implicating sub-optimal practice. Our future work is to expand our qualitative analysis on more sample size and to show that the sub-optimal practice of smell addition or removal contributes to the high probability of the post-release defect. In our study, we hope to fill the gap between academia and industry by providing evidence of sub-optimal practice in the way we address test smells, and how they may be detrimental to the software.

REFERENCES

[1] Mark Aberdour. 2007. Achieving quality in open-source software. *IEEE software* 24, 1 (2007), 58–64.

[2] Dimitrios Athanasiou, Ariadi Nugroho, Joost Visser, and Andy Zaidman. 2014. Test code quality and its relation to issue handling performance. *IEEE Transactions on Software Engineering* 40, 11 (2014), 1100–1125.

[3] Gabriele Bavota, Abdallah Qusef, Rocco Oliveto, Andrea De Lucia, and Dave Binkley. 2015. Are test smells really harmful? An empirical study. *Empirical Software Engineering* 20, 4 (2015), 1052–1094.

[4] Martin Fowler and Matthew Foemmel. 2006. Continuous integration. *Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf* 122 (2006), 14.

[5] Vahid Garousi and Baris Küçük. 2018. Smells in software test code: A survey of knowledge in industry and academia. *Journal of Systems and Software* 138 (2018), 52–81. https://doi.org/10.1016/j.jss.2017.12.013

[6] Mary Jean Harrold. 2000. Testing: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*. ACM, 61–72.

[7] Fabio Palomba and Andy Zaidman. 2017. Does Refactoring of Test Smells Induce Fixing Flaky Tests?. In *2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017, Shanghai, China, September 17-22, 2017*. 1–12. https://doi.org/10.1109/ICSME.2017.12

[8] Fabio Palomba and Andy Zaidman. 2019. The smell of fear: On the relation between test smells and flaky tests. *Empirical Software Engineering* (2019), 1–40.

[9] Anthony Peruma, Khalid Almalki, Christian D. Newman, Mohamed Wiem Mkaouer, Ali Ouni, and Fabio Palomba. 2019. On the Distribution of Test Smells in Open Source Android Applications: An Exploratory Study. In *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering (CASCON '19)*. IBM Corp., Riverton, NJ, USA, 193–202. http://dl.acm.org/citation.cfm?id=3370272.3370293

[10] Gregg Rothermel, Roland H. Untch, Chengyun Chu, and Mary Jean Harrold. 2001. Prioritizing test cases for regression testing. *IEEE Transactions on software engineering* 27, 10 (2001), 929–948.

[11] Davide Spadini, Fabio Palomba, Andy Zaidman, Magiel Bruntink, and Alberto Bacchelli. 2018. On the relation of test smells to software code quality. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 1–12.

[12] Nikolaos Tsantalis, Matin Mansouri, Laleh M Eshkevari, Davood Mazinanian, and Danny Dig. 2018. Accurate and efficient refactoring detection in commit history. In *Proceedings of the 40th International Conference on Software Engineering*. ACM, 483–494.

[13] Arash Vahabzadeh, Amin Milani Fard, and Ali Mesbah. 2015. An empirical study of bugs in test code. In *2015 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 101–110.

[14] Arie Van Deursen, Leon Moonen, Alex Van Den Bergh, and Gerard Kok. 2001. Refactoring test code. In *Proceedings of the 2nd international conference on extreme programming and flexible processes in software engineering (XP2001)*. 92–95.