Practitioner Perceptions of Ansible Test Smells

Yue Zhang* Fan Wu† Akond Rahman§
*Department of Computer Science, Tuskegee University, Tuskegee, AL, USA
§Department of Computer Science and Software Engineering, Auburn University, Auburn, AL, USA
Email: *yzhang8317@tuskegee.edu †fwu@tuskegee.edu §akond@auburn.edu

***Abstract*—The practice of infrastructure as code (IaC) recommends automated management of computing infrastructure with application of quality assurance, such as linting and testing. To that end, researchers recently have investigated quality concerns in IaC test manifests by deriving a catalog of test smells. The relevance of the identified smells need to be quantified by obtaining feedback from practitioners. Such feedback can help the IaC community understand if smells have relevance amongst practitioners, and derive future research directions. We survey 30 practitioners to assess the relevance of three Ansible test smell categories namely, assertion roulette, local only testing, and remote mystery guest. We observe local only testing to be the most agreed upon test smell category, whereas, assertion roulette is the least agreed upon test smell category. Our findings provide a nuanced perspective of test smells for IaC, and lays the groundwork for future research.**

***Index Terms*—ansible, devops, empirical study, infrastructure as code, testing, smells**

I. Introduction

With the advent of cloud computing resources, automation tools, such as Ansible have gained in popularity amongst organizations. Ansible is used to implement infrastructure as code (IaC), which is the practice of automatically managing computing infrastructure at scale. The practice of IaC advocates for integration of recommended software engineering practices, such as quality assurance so that IaC manifests, such as Ansible manifests do not include defects. Prior research [3], [4], [9] has documented defects in Ansible manifests. If defects remain unmitigated, then these defects can lead to large-scale consequences [3]. Accordingly, researchers have advocated for integration of quality assurance for IaC manifests used to provision computing infrastructure, and also for test manifests used to test the provisioning process.

Recently, researchers [7] have focused on improving quality assurance for Ansible test manifests. They [7] have derived a list of test smells, i.e., coding patterns in test manifests that are correlated with defects. They [7] identified three test smell categories: assertion roulette, local only testing, and remote mystery guest. One limitation of the researchers' work [7] is related to practitioner validation: they did not quantify practitioners' perceptions of the identified test smell categories. Addressing such limitation is important as practitioner agreement with the identified test smell categories can provide validation and evidence of relevance for the identified test smell categories. Practitioner-reported disagreements can reveal the reasons for disagreements, and how test smell-related research can be improved by considering practitioner context.

Accordingly, we answer the following research question: ***RQ**: How do practitioners perceive test smell categories for Ansible test manifests?*

We conduct a survey with 30 practitioners to quantify the relevance of test smells documented for Ansible manifests. **Our contribution**: is quantification of practitioner perceptions for Ansible test smell categories.

II. Related Work

Our paper is related to prior research that have investigated test smells for general purpose programming languages (GPLs), such as Java. Van Deursen et al. [16] identified 11 categories of test smells that can be refactored using 6 activities. Tufano et al. [15] built on top of van Deursen et al. [16]'s work to investigate how practitioners perceive 5 categories of JUnit test smells. Qusef et al. [8] proposed a technique to predict the presence of faults in production source code by leveraging metrics related to test smells. Junior et al. [12] reported that practitioners' professional experience is not a root cause for introducing test smells. Spadini et al. [14] investigated JUnit test cases collected from 10 projects, and reported that test cases with test smells are more likely to be more defect-prone. Bavota et al. [1] reported test smells to negatively impact software maintenance. In another paper, Bavota et al. [2] quantified the relationship between test smells and test code comprehension, and observed that in the absence of test smells comprehension is 30% better. Practitioner perceptions of test smells have also been investigated. For example, Junior et al. [12] surveyed 60 practitioners to understand whether test professionals inadvertently insert test smells, and reported that experienced practitioners frequently introduced test smells. In another paper, Tufano et al. [15] surveyed 19 practitioners to investigate practitioners' perceptions of 5 categories of JUnit test smell, and reported that practitioners generally do not perceive test smells as actual problem. Tufano et al. [15] also suggested that automated tools for identifying test smells are needed for practitioners.

While there is a plethora research related to test smells for GPLs, we observe a lack of characterizing the practitioner perceptions of test smells for Ansible manifests. We address this gap in our paper.

TABLE I: Name, Definition, and Smell Density of Identified Test Smells Documented by Hassan and Rahman [7].

Smell Name	Definition	Smell Density
Assertion Roulette	The recurring coding pattern of using multiple undocumented assert statements, which is unable to identify a specific failed assertion.	3.16
Local Only Testing	The recurring coding pattern of executing tests in the local development environment only.	1.47
Remote Mystery Guest	The recurring coding pattern of using external resources.	4.59

III. Methodology

We answer RQ by conducting an online survey with practitioners who develop IaC manifests. We first ask practitioners about their experience in developing IaC manifests. Next, we ask how frequently practitioners test IaC manifests using a Likert scale: never, rarely, about half of the time, most of the time, and always. Then, we ask "*We have identified three test smell categories by analyzing open source test manifests used for IaC. Each of these categories are listed below. To which extent do you agree that these test smell categories are applicable for IaC?*". The definitions of each test smell category is provided in Table I with smell density reported by Hassan and Rahman [7]. Smell density is the count of test smells that appear in every 1,000 lines of test code for Ansible manifests [7]. For each smell category we provide names, definitions, and examples. We construct the survey following Kitchenham and Pfleeger's guidelines [6]: (i) to measure agreement use of a Likert scale: strongly disagree, disagree, neutral, agree, and strongly agree; (ii) add explanations related to study purpose, (iii) provide an estimate of completion time; and (iv) conduct a pilot survey to get initial feedback. From the feedback of the pilot survey, we add an open-ended question so that survey respondents can provide more context including feedback on the reasons they agreed or disagreed. The survey questionnaire is available online [19].

We conduct the survey using e-mails. We obtain e-mail addresses by mining repositories used in prior work [7]. We randomly select 250 e-mail addresses, which we use to send the survey link. We offer a drawing of one 50 USD Amazon gift card as an incentive for participation following Smith et al. [13]'s recommendations. We conduct the survey from February 20, 2021 to October 20, 2022 following the Internal Review Board (IRB) protocol #2356.

IV. Survey Results

We obtain 30 responses (response rate=12.0%). We acknowledge the survey response rate to be low, which is common in software engineering research. Smith et al. [13] reported software engineering survey response rate to vary from 6% to 36%. A breakdown of respondents' reported experience with IaC is reported in Table II. Table III provides a breakdown of reported test frequency. We observe 76.6% of survey respondents to either 'never' or 'rarely' test IaC manifests.

We provide our answer to RQ in Figure 1, where we provide the agreement rate for each identified category. The right hand side provides the percentage of survey participants who agreed or strongly agreed with the identified test smell categories. The left hand side of Figure 1 provides the percentage of participants who disagreed or strongly disagreed with the test smell categories.

TABLE II: Survey Respondents' Experience in IaC

Experience	Respondent count
$1-2$ years	14
$>2-3$ years	2
$>3-4$ years	4
$>4-5$ years	3
>5 years	7

TABLE III: Testing Frequency Reported by Respondents

Reported Frequency	Respondent count
Never	8
Rarely	15
About half of the time	2
Most of the time	4
Always	1

According to survey results, the most agreed upon category is local only testing. The least agreed upon smell category is assertion roulette. We observe an agreement rate of 50% or more for local only testing and remote mystery guest.

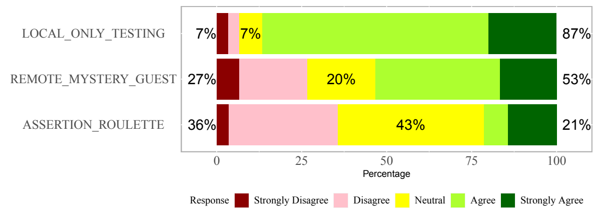

Fig. 1: Feedback from practitioners: the most agreed upon test smell category is local only testing.

We identify insights on practitioner perceptions related to identified smell categories. One practitioner agreed with local only plays stating "*Testing on a local workstation is unlikely to be representative or meaningful. One big example is that developer workstations are usually heavily loaded with many libraries, runtimes, development tools and SDKs, which may be assumed by the Ansible script but not present on a minimal server install*". We also document disagreements. One practitioner disagreed with mystery guest as a test smell category stating " *I don't see anything bad in using external dependency to do some checks. It all depends on the size of the dependency and the amount of dependencies*". A practitioner disagreed with assertion roulette explaining "*While it can be abused, compound assertions are sensible when they are clear and logically related*".

V. Discussion and Conclusion

Section IV provides a nuanced practitioner perspective on identified test smell categories. We observe assertion roulette is the least agreed upon test smell category, even though for GPLs assertion roulette is positively correlated with the existence of faults [8], and defect proneness [14]. Furthermore, unlike GPLs for which assertion roulette is the most agreed upon test smell category [15], assertion roulette for Ansible test manifests is the least agreed upon category. One possible explanation can be attributed to IaC manifests' use of domain specific languages (DSLs) [10], [11]. The syntax and semantics of DSLs are different from GPLs [5], [17], [18]. Our hypothesis is that differences between DSLs and GPLs can correlate with the characterization of test smells in Ansible manifests.

We also observe remote mystery guest to be the second least agreed upon category, even though it is the most frequent category. One possible explanation is that practitioners are not aware of test smell consequences and presence, and inadvertently include such smells in test plays due to a lack of awareness. Tufano et al. [15] also reported practitioners to be unaware of test smells in JUnit test cases.

Conclusion: Our empirical study showcases that not all identified test smells for Ansible manifests have relevance amongst practitioners. For example, the least agreed upon test smells category for Ansible is assertion roulette, which is the second most frequent test smell category based on smell density [7]. The implication of our finding is that in future work, researchers can investigate why test smells have varying practitioner perceptions, and use the gathered explanations to better design test-related linting tools. Based on our findings, we conclude that the test smell taxonomy derived by Hassan and Rahman [7] lays the groundwork to further understand the context of why test smells appear in Ansible manifests, which can be incorporated into a test smell detection tools.

Acknowledgments

We thank the PASER group at Auburn University for their valuable feedback. This research was partially funded by the U.S. National Science Foundation (NSF) Award # 2247141, Award # 2310179, Award # 2209637, and the U.S. National Security Agency (NSA) Award # H98230-21-1-0175.

References

[1] G. Bavota, A. Qusef, R. Oliveto, A. De Lucia, and D. Binkley, "An empirical analysis of the distribution of unit test smells and their impact on software maintenance," in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, 2012, pp. 56–65.

[2] G. Bavota, A. Qusef, R. Oliveto, A. De Lucia, and D. Binkley, "Are test smells really harmful? an empirical study," *Empirical Software Engineering*, vol. 20, no. 4, pp. 1052–1094, 2015.

[3] S. Dalla Palma, D. Di Nucci, F. Palomba, and D. A. Tamburri, "Within-project defect prediction of infrastructure-as-code using product and process metrics," *IEEE Transactions on Software Engineering*, vol. 48, no. 6, pp. 2086–2104, 2022.

[4] ——, "Toward a catalog of software quality metrics for infrastructure code," *Journal of Systems and Software*, vol. 170, p. 110726, 2020.

[5] P. Hudak, "Modular domain specific languages and tools," in *Proceedings. Fifth International Conference on Software Reuse (Cat. No.98TB100203)*, Jun 1998, pp. 134–142.

[6] B. A. Kitchenham and S. L. Pfleeger, *Personal Opinion Surveys*. London: Springer London, 2008, pp. 63–92. [Online]. Available: https://doi.org/10.1007/978-1-84800-044-5_3

[7] H. Mohammad Mehedi and A. Rahman, "As code testing: Characterizing test quality in open source ansible development," in *2022 15th IEEE Conference on Software Testing, Verification and Validation (ICST)*. Los Alamitos, CA, USA: IEEE Computer Society, apr 2022. [Online]. Available: https://akondrahman.github.io/publication/icst2022

[8] A. Qusef, M. O. Elish, and D. Binkley, "An exploratory study of the relationship between software test smells and fault-proneness," *IEEE Access*, vol. 7, pp. 139 526–139 536, 2019.

[9] A. Rahman, E. Farhana, C. Parnin, and L. Williams, "Gang of eight: A defect taxonomy for infrastructure as code scripts," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 752–764. [Online]. Available: https://doi.org/10.1145/3377811.3380409

[10] A. Rahman, E. Farhana, and L. Williams, "The 'as code'activities: development anti-patterns for infrastructure as code," *Empirical Software Engineering*, vol. 25, no. 5, pp. 3430–3467, 2020.

[11] A. Rahman and L. Williams, "Characterizing defective configuration scripts used for continuous deployment," in *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, April 2018, pp. 34–45.

[12] N. Silva Junior, L. Rocha, L. Almeida Martins, and I. Machado, "A survey on test practitioners' awareness of test smells," *arXiv*, pp. arXiv–2003, 2020.

[13] E. Smith, R. Loftin, E. Murphy-Hill, C. Bird, and T. Zimmermann, "Improving developer participation rates in surveys," in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, May 2013, pp. 89–92.

[14] D. Spadini, F. Palomba, A. Zaidman, M. Bruntink, and A. Bacchelli, "On the relation of test smells to software code quality," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2018, pp. 1–12.

[15] M. Tufano, F. Palomba, G. Bavota, M. Di Penta, R. Oliveto, A. De Lucia, and D. Poshyvanyk, "An empirical investigation into the nature of test smells," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE 2016. New York, NY, USA: Association for Computing Machinery, 2016, p. 4–15. [Online]. Available: https://doi.org/10.1145/2970276.2970340

[16] A. Van Deursen, L. Moonen, A. Van Den Bergh, and G. Kok, "Refactoring test code," in *Proceedings of the 2nd international conference on extreme programming and flexible processes in software engineering (XP)*, 2001, pp. 92–95.

[17] E. Van Wyk, L. Krishnan, D. Bodin, and A. Schwerdfeger, "Attribute grammar-based language extensions for java," in *Proceedings of the 21st European Conference on Object-Oriented Programming*, ser. ECOOP'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 575–599. [Online]. Available: http://dl.acm.org/citation.cfm?id=2394758.2394796

[18] M. Voelter, *DSL Engineering: Designing, Implementing and Using Domain-Specific Languages*. USA: CreateSpace Independent Publishing Platform, 2013.

[19] Y. Zhang and A. Rahman, "Verifiability package for paper," https://figshare.com/s/84e12acb3f5c9a2e0af2, 2022, [Online; accessed 01-Dec-2022].