A survey on test practitioners' awareness of test smells

Nildo Silva Junior, Larissa Rocha, Luana Almeida Martins, and Ivan Machado

Federal University of Bahia - UFBA, Salvador - BA, Brazil
nildo.silva@ufba.br, larissars@dcc.ufba.br, martins.luana@ufba.br,
ivan.machado@ufba.br

Abstract. Developing test code may be a time-consuming task that usually requires much effort and cost, especially when it is done manually. Besides, during this process, developers and testers are likely to adopt bad design choices, which may lead to the introduction of the so-called test smells in test code. Test smells are bad solutions to either implement or design test code. As the test code with test smells increases in size, these tests might become more complex, and as a consequence, much harder to understand and evolve correctly. Therefore, test smells may have a negative impact on the quality and maintenance of test code and may also harm the whole software testing activities. In this context, this study aims to understand whether test professionals non-intentionally insert test smells. We carried out an expert survey to analyze the usage frequency of a set of test smells. Sixty professionals from different companies participated in the survey. We selected 14 widely studied smells from the literature, which are also implemented in existing test smell detection tools. The yielded results indicate that experienced professionals introduce test smells during their daily programming tasks, even when they are using standardized practices from their companies, and not only for their personal assumptions. Another relevant evidence was that developers' professional experience can not be considered as a root-cause for the insertion of test smells in test code.

Keywords: Test smells · Empirical study · Expert Survey.

1 Introduction

Software projects, both commercial and open source, commonly encompass a set of automated test suites, as a crucial support to verify software quality [4]. However, creating test code may require high effort and cost [4, 20]. For example, the SCADA project used 870 hours to create a set of test scripts [19]. Automated test generation tools, such as Randoop[1], JWalk[2], and Evosuite[3], emerge as alternatives to facilitate and streamline this process. If designed with high quality,

---

[1] https://randoop.github.io/randoop/

[2] http://staffwww.dcs.shef.ac.uk/people/A.Simons/jwalk/

[3] http://www.evosuite.org/

automated testing offers benefits over manual testing, such as repeatability, predictability, efficient test runs, and thereby require less effort and costs [5, 20]. In this way, tests should be concise, repeatable, robust, sufficient, necessary, clear, efficient, specific, independent, maintainable, and traceable [10].

However, the development of a well-designed test code is not a straightforward task. Developers are usually under time pressure and must deal with constrained budgets, which can stimulate the use of anti-patterns in test code, leading to the occurrence of the so-called test smells. Test smells are identifiers of poor implementation solutions and problems in test code design [6]. As a consequence of introducing test smells, the produced test code would likely have reduced quality and, consequently, may not reach its expected capabilities at finding bugs while being understandable, maintainable, etc. [5, 20]. The literature reports 196 test smell types, classified in the following groups: behavior, logic, design related, issue in test steps, mock and stub-related, association in production code, code-related, and dependencies [5].

Some studies have been conducted to identify and analyze the effect of the presence of test smells in software projects in several respects [4, 6, 12, 17]. In those studies, the test smells were presented as non-functional quality attributes within the Software Test Code Engineering process, for example. Some test smell types are presented, and there is a discussion on what they may cause in the test code maintenance [4]. Metrics were defined to identify test smells on automated tools and these tools were validated by developers [6] or through a case study [17]. Some test smells were also analyzed as a way to reduce flaky tests, i.e. tests with non-deterministic behavior [12]. However, discussions about daily practices and programming styles that may contribute to inserting test smells are still lacking. Understanding the relationship between development practices and test smell insertion may support improving the test creation process.

The study aims to understand whether professionals non-intentionally insert test smells. Through an expert survey [9], we could analyze the practices adopted by developers that might introduce test smells and how frequently these practices are used during test creation and observed during test execution. The survey counted on sixty participants who work for different Brazilian companies. Our analysis may provide insights towards a better understanding of how and which practices may lead to the insertion of test smells in test code.

The following research questions were addressed in this study:

- **RQ1: Do professionals use test case design practices that might lead to test smell insertion?** We investigate whether bad design choices may be related to test smells.
- **RQ2: Does the professional experience interfere with test smell insertion?** We investigate whether, over time, professionals improve the test creation process.
- **RQ3: Which are the practices present in professionals' daily activities that lead test smells insertion?** We investigate which test smells are associated with the most frequent professionals' practices.

2 Test Smells

Automated tests may yield more efficient results when compared to manually executed ones. Due to their capability of being easily repeated several times, and the lack of human interference, automated tests might lead to reductions in both time and execution effort [5, 20]. However, developing test code is not a trivial task, and once it has been poorly designed, the use of automated tools may not ensure the quality of the system [11, 18]. As aforementioned, in real-world practice, developers are likely to use anti-patterns during test creation and evolution, which may lead to mistakes in test code implementation [1, 16]. These anti-patterns may negatively impact the maintenance of the test code [17].

Several studies have investigated different types of smells. Initially, Deursen et al. [16] defined a catalog of 11 test smells and refactorings to remove test smells from test code. After that, many other authors extended this catalog and analyzed the effects of the smells on both production and test code [1, 2, 4, 6, 10–13, 16–18]. For example, Garousi and Küçük [5] found more than 190 test smells on a literature review of 166 studies.

In this study, we selected 14 types of test smells, which are frequently studied and implemented in cutting-edge test smell detection tools [10, 13, 16]. These are described next:

- **Assertion Roulette (AR)**. It occurs when tests present assertions with no explanation in test methods. If one of those assertions fails, it is not possible to identify which one caused the problem;
- **Conditional Test Logic (CTL)**. It occurs when tests present conditional logic (if-else or repeat instructions). Tests with this structure do not guarantee that the same flow is verified, besides allowing a particular piece of code not to be tested;
- **Constructor Initialization (CI)**. This smell occurs when test methods present a constructor;
- **Eager Test (ET)**. It occurs when a test method checks many object methods at the same time. This test may be hard to understand, and to execute;
- **Empty Test (EpT)**. Occurs when test methods do not contain executable assertions;
- **For Testers Only (FTO)**. Occurs when a production class has methods only used by test methods;
- **General Fixture (GF)**. It occurs when the configuration file is generic, and different tests perform tests using part of configuration data. Those files may be hard to read, understand, and they may slower test execution;
- **Indirect Testing (IT)**. It occurs when test class has methods that perform tests in different objects because there are references to those objects at the test class, for example.
- **Magic Numbers (MN)**. Occurs when tests present a literal number as a test parameter;
- **Mystery Guest (MG)**. This smell occurs when the test uses an external resource, such as a file with test data. If the external file is removed, the test results may fail;

- **Redundant Print (RP)**. Occurs when test methods contain irrelevant print statements;
- **Resource Optimism (RO)**. It occurs when the test code contains optimist assumptions about the presence or absence of external resources. The test may return a positive result once, but it may fail for the next times;
- **Test Code Duplication (TCD)**. Occurs when test code has undesired duplication;
- **Test Run War (TRW)**. It occurs when a test passes only when it is performed isolated and fails when it is performed with another test at the same time.

3   Research Methodology

3.1   Survey Design

For this study, we used the design of observation by case control. It is a descriptive design used to investigate previous situations to support the understanding of a current phenomenon [7]. We are interested in detecting the most common practices that may introduce test smells. In the questionnaire, we did not use the *test smell* term or any of its synonymous not to influence the respondents. Instead, we transcribed the rationale for each test smell and converted them into practices for both test creation and execution. Once the respondents claim they commonly adopt a given practice, and the practice is the definition of a test smell, they assume that they either insert (during test creation) or identify (during test execution) a test smell. Table 1 shows examples of those practices.

The questionnaire was split into three blocks. The first one leverages the respondents' profile. It encompasses thirteen questions, aimed to identify respondents' age, gender, degree, and testing and programming skills. The second one encompasses fourteen statements and six complementary questions: four categorical objective questions and two open-ended questions. The statements describe practices related to test smells. The respondents chose one out of five possible answers (*always, frequently, rarely, never*, and *not applicable*) for each statement, where *always* indicates the adoption of bad practice for test creation. For example, the statement *"I've already created a test to validate a feature that would not be used in the production environment"* corresponds to the `For testers only` smell. Whether the answer was *"Always"*, it meant the respondent usually uses that practice in his daily tasks, and therefore it is likely he commonly inserts this smell in the test code he develops. The six complementary questions are designed to understand how the professionals deal with the test creation process.

The third block is similar to the second one. However, there are fourteen statements taking the perspective of who executes the tests, in which a high response rate (i.e., *Always*) indicates that a respondent usually come across tests with test smells; besides, there is one complementary open-ended question, aimed to understand which problems the professionals are likely to deal with when executing the tests.

Table 1. Examples of practices related to test smells.

Test Smell	Practices for Test Creation	Practices for Test Execution
Mystery Guest	I often create test cases using some configuration file (or supplementary) as support	A test case fails due to unavailability of access to some configuration file.
Eager Test	I often create tests with a high number of parameters (number of files, database record, etc.).	I run some tests without understanding what its purpose is.
Assertion Roulette	I pack different test cases into one (i.e., put together tests that could be run separately).	Some tests fail and it is not possible to identify the failure cause.
For Testers Only	I have already created a test to validate some feature that will not be used in the production environment	I run some tests to validate features that will not be used in the production environment.
Conditional Test Logic	I have already created conditional or repeating tests.	I run tests with conditional or repeating structure.
Empty Test	I have already created an empty test, with no executable statement.	I find empty test, with no executable statement.

It is worth mentioning that we executed a pilot survey with 4 professionals to identify improvement opportunities. Based on their responses, we could improve the questionnaire prior to running the actual survey.

The questionnaire used in this survey is available online[4]. Data were gathered from April 3rd to June 3rd in 2019.

3.2 Data analysis

To answer RQ1, we analyzed the objective questions about test creation and test execution related to test smells. Each question encompassed an affirmative statement aimed to describe daily practices in software testing, such as *"I usually found empty test"* and *"I usually create test with part of external configuration file"*.

For RQ2, we compared the professional experience with the frequency of the use of test smells. We also used the same answer format as RQ1, but only considered the test creation process. During the test execution process, professionals identify test smells instead of creating them.

To answer RQ3, we grouped the practices by frequency to identify which ones are most commonly used. The practices may be associated to one or more test

---

[4] A copy of the questionnaire is available online at `https://bit.ly/2RQVDdc`

smells according to their characteristics, such as external file usage, conditional structure, and programming style.

The three open questions were analyzed through coding and continuous comparison [8]. Our main intention was to understand the reasons that might lead the software engineers to use practices that may insert test smells. Besides, we were also intended to understand which difficulties they come across when creating and executing tests. The coding task was performed by two researchers and validated by consensus. We also associated some of the practices with the test code characteristics defined by Meszaros et al. [10].

Since the questionnaire encompasses both close-ended and open-ended questions, after data collection, we employed open coding to identify additional reasons on why professionals use bad practices on testing activities. Codes were peer-reviewed and changed upon agreement with the authors of this paper. Considering that open questions were optional, we used coding to complement our results.

4   Results

4.1   Participants

We sent the invitation to professionals from eight Brazilian companies in a convenience sampling. We created one copy of the questionnaire for each of these companies (C1-C8) and sent them by email. The different versions of the questionnaire served to control the number of respondents from each company. Those companies have from 4 to 66 testing professionals, who perform both manual and automated tests, as Table 2 shows.

Besides, we also sent the questionnaire by direct message (D1) and posted it on a Facebook group dedicated to discussing software testing (G1). In total, 305 professionals were contacted.

As a result, we received 60 answers[5] (out of 305 potential respondents) from three different states in Brazil: BA (66.7%), SP (31.7%), and PR (1.6%). The respondents were from 22 to 41 years old, and their experience with quality assurance ranged from 0 to 13 years (5.16 on average). The experience as software developers also ranged from 0 to 13 years (average 1.67). Most of the respondents were male (65%), but results also showed female (32%) and non-binary gender (3%). Most of them hold a degree in Computer Science-related courses (83.3%), some of them hold a degree in other STEM courses (10%), and some of them hold a degree in other areas (6.7%). 90% of the respondents pursued higher education degrees, as follows: 66.7% hold a bachelor's degree, 21.7% hold a graduate degree, and 1.6% hold a postdoc.

Regarding their commonly performed software testing tasks, most of the respondents reported creating and running tests at the same rate (43.3%); but also executed tests with more frequency than they created (21.7%); and created tests with more frequency than they executed (13.3%). Moreover, some respondents only executed test cases (20%); and other respondents only created

_____

[5] Raw data with all the answers (PT-BR) are available at `https://bit.ly/33ZOESi`

Table 2. respondents

Source	Professionals	Answers	Source	Professionals	Answers
C1	66	14	C6	4	4
C2	30	1	C7	4	4
C3	10	0	C8	4	0
C4	6	0	D1	52	35
C5	5	0	G1	124	2

them (1.7%). They performed tests over many different platforms, and most of them (53%) worked with two or more platforms: Web (65%), Android (58%), Desktop (48%), and Apple (28%) applications. Backend, microservices, API, mainframe, and cable TV had 1.67% each.

Usually, professionals tested solutions for mobile device applications (65%) and web applications (60%). We also identified the following domains they work with: embedded systems (23,33%), cloud computing (18,33%), information security (11,67%), internet of things (6,67%), big data, retail, artificial intelligence, cable tv, bioinformatics, commercial information, desktop system, and payment solutions (1,67% each one). The sources of test creation were requirements (48%); developed software (45%); source code and use cases (2% each); and 3% did not know the source.

4.2   Test creation and execution practices

Gathered data made it possible to establish a relationship between test creation and execution practices and the occurrence of test smells. Figure 1 summarizes such a relationship, by showing the frequency of test smells usage during (a) test creation and (b) test execution processes, based on the responses. We next discuss the main results.

The search for test duplication was a personal practice for most of the respondents (48%). In some cases, it was also a company practice (18%), or even both, respondent and company practices (6%). However, for 28% of respondents, this activity was not applied. Checking tests with the same objective reduces the **Test Duplication** smell.

During the test creation process, **Conditional Test Logic** and **General Fixture** were the most frequent test smells. The former obtained 47% of *Always* and *Frequently* answers, and the latter, 45% in both answers, as Figure 1(a) shows. A high rate of those responses may indicate a common use of practices related to those smells. We also analyzed why developers create tests with bad practices (one open non-mandatory question answered by 45% of respondents). The main reasons are related to company or personally employed standards, limited time, and attempt to reach better coverage and efficiency.

We also asked whether they have already changed existing test sets because they found the bad practices we presented. 12% of respondents always perform any changes in the test code, when they find test smells-related practices; 38%
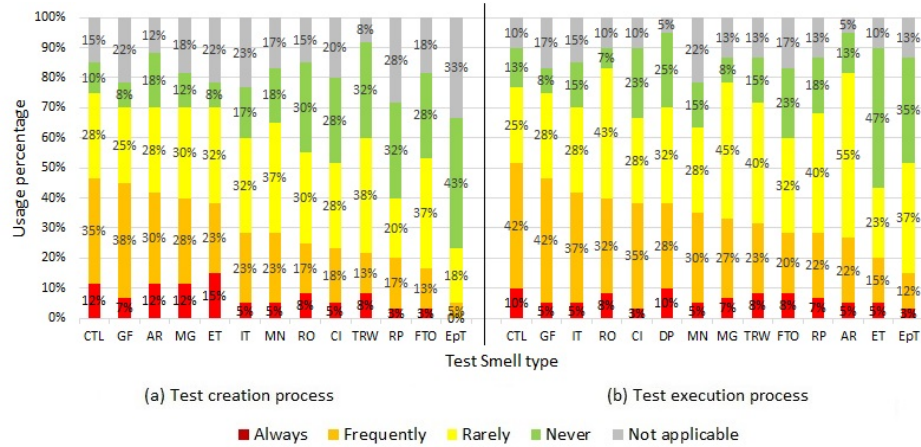
Fig. 1. Test Smells frequencies on test creation and execution.

frequently change them; 27% rarely; 12% never edit test code, and 11% answered as not applicable. Among the reasons to change the test, they informed that changing it reduces ambiguities (30%); improves execution speed (27%); they need to adequate it to company standards (23%); they change it when they do not understand the test purpose (13%); and when the corresponding production class evolves (7%).

Furthermore, the respondents had to point out which test structure problems they face. The results indicated that some tests depended on third party resources (52%); were hard to understand (48%); contained either unnecessary information or were ambiguous (40% each); depended on external files (33%); used external configuration file (10%); and presented resources limitation (2%).

Regarding difficulties in creating test cases (one open non-mandatory question answered by 38% of respondents), requirement issues were the most frequent ones (56%). Other problems were related to difficulties in performing test code reuse, lack of knowledge, issues in production code, code coverage, test environment problems, and time and resources limitation.

The questions regarding test execution also presented a sequence of statements about ordinary situations the developers usually face, in which respondents should answer according to frequency. **Conditional Test Logic** (52%) and **General Fixture** (47%) were the test smells most cited during test execution, as Figure 1(b) shows.

The most cited problem related to test execution refers to difficulties involving the test environment (34%). Some of the cited issues were related to the unavailability of the test environment, the demand for 3rd party features, or even low-performance environments. The second most common problem is associated with understanding the test purpose (28%). They informed that tests are poorly written, which allow multiple interpretations, besides lack of writing
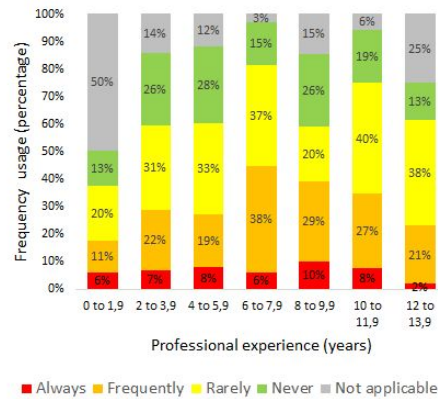
Fig. 2. Test Smells frequencies on test creation according professional experience.

standard. Lack of test maintenance was the third problem (24%), which involves outdated and incomplete tests due to the evolution of the system code.

4.3 Professional Experience

Although most respondents reported they create and execute tests at the same proportion, our investigation presents a different scenario as the tester gets more experienced. For the first two years of experience, there was a predominance of test execution. 64% only executed tests; 9% frequently executed tests and occasionally created them, and 27% created and executed them at the same proportion. Over the years, mainly after 10 years of experience, practitioners tend to create more tests than execute them. We found that the ones who only created tests had at least 12 years of professional experience. Thus, the less experienced the tester, the less number of tests they create and the more tests they execute.

We also analyzed the use of good practices to create tests as professionals become more experienced. During the creation of a test, the use of good practices tends to increase over the years, as *Rarely* and *Never* got more responses than the remainder. However, gathered data showed a slight reduction between eight and ten years, but it grows back from ten years of experience. On the other hand, the use of bad practices also increases over the years, with *Always* and *Frequently* gaining more responses, to respondents with up to eight years of experience, and decreasing thereafter, as Figure 2 shows.

5 Discussion

5.1 Test design practices that lead to test smells insertion (RQ1)

We observed that some practices might lead to the introduction of test smells in test code. Therefore, we analyzed the frequency of those practices for test creation and execution.

In the former, we observed that every test smell presented at least three out of four possible answers (*Always, Frequently, Rarely, and Never*). We then classified such gathered data into two groups: Commonly-used practices group (CPG) and the Unused practices group (UPG). CPG contains test smells that mostly present *Always* and *Frequently* as answers, and UPG that mostly present *Rarely* and *Never* as answers. We consider a test smell belonging to one of the groups when the difference between the rates of *Always* and *Frequently* versus *Rarely* and *Never* is greater than 10%. For example, `Empty Test, For testers only, Test run War, Constructor Initialization, Resource Optimism, Redundant Print, Magic Number, Indirect test` belong to the UPG group, which means professionals rarely insert those smells on testing activities. Conversely, professionals frequently adopt practices related to `General Fixture`, the only member of the CPG group. Thus, it is possible that they create many tests with that smell, which may compromise test maintenance [15]. Still, four test smells presented a similar frequency of pertinence to both groups (less than 10% of difference). For them, there was not a pattern among respondents. For instance, the `Eager Test` smell obtained 38% to CPG and 40% to UPG.

In the latter, UPG contains the following test smells: `Empty Test, Eager Test, Assertion Roulette, Redundant Print, Duplicated Test, Test Run War, For testers Only, Mystery Guest, Constructor Initialization, and Resource Optimism`, which means that professionals rarely face those smells during testing. On the other hand, professionals frequently find practices related to two test smells, which are part of CPG: `General Fixture` and `Conditional Test Logic`. Besides, other two test smells presented a similar frequency of pertinence to both groups, `Indirect Test` and `Magic Number`. Thus, there is no perceived standard among respondents for them. In general, our study identified that all test smells we analyzed appear in the testing activities. They all were cited by respondents, even rarely.

We also analyzed the reasons that lead professionals to adopt the practices presented in the survey. Since this data came from open-ended questions, we identified 16 different codes, in which the most common ones are: *company standard, personal standard, project politics, professional experience, saving time* and *improving coverage*. For example, professional #26 reported applying company standards when creating tests, and those standards may insert smells. He said he commonly applies bad practices *"to match company development standards."* Otherwise, professional #54 reported personal standard when said that *"I group tests by module so that tests can be executed sequentially without compromising effectiveness."* This behavior also indicates that participants may have misunderstood the definition of test smells. When grouping tests, it is possible to insert the `Assertion Roulette` test smell and to compromise test independence. Similar situation occurred to professionals #14, #16, #27, #50 and #59.

Table 3 summarizes the answers for each RQ.

Table 3. Summary of answers for each research question.

RQ	Answer
RQ1	We found that professionals do adopt practices for test case design, which introduce test smells. Usually, those practices come from improper personal and company standards.
RQ2	Experienced software testing professionals may not produce less test smells than inexperienced ones.
RQ3	The practices most present in the daily life of professionals that lead to test smells insertion were the use of conditional structure or repetition and the use of generic configuration data.

5.2 Professional experience and its interference on test smell insertion (RQ2)

Although we analyzed the experience of practitioners and its influence on the adoption of practices that introduce test smells, we have not identified any clear correlation. For example, the *Always* option indicates they always use bad practices. Regarding the test creation process, we cannot infer that inexperienced professionals introduce more smells on tests than the experienced ones. We found the following rates (from less experienced to the most experienced professionals): 6% of professionals from 0-2 work years answered *Always*, 7% to 2-4 work years, 8% to 4-6, 6% to 6-8, 10% to 8-10, 8% 10-12 and 2% for over 10 years.

When testers are inexperienced programmers, they may write lower quality tests, but when they are more experienced, they can carry programming biases that may also contain bad practices. The absence of a tendency indicates a non-behavioral change between less and more experienced software testing professionals.

5.3 Professional practices that might lead to introducing test smells (RQ3)

Tests can be performed either manually or automatically. Although there are specific tools to support test automation [3, 14], 62% of respondents perform more manual than automated tests. In addition, they are also inexperienced in software development; 55% have no experience with software development (less than 2 years of experience on average). Lack of experience with software development in general may contribute to the use of bad practices.

According to the practices explored in this study, we identified that two development activities are very present in professionals' daily life: *(ii) the use of generic configuration data*, which produces `General Fixture`. This smell is the most frequent on test creation and execution processes for CPG; and *(i) the use of conditional structure or repetition*, which is directly associated to

`Conditional Test logic` smell. It was the second most detected smell on test execution (CPG group).

Professionals indicated they commonly face several problems with tests, such as poorly written tests and outdated and incomplete test procedures. According to them, when tests are associated with generic configuration data, test case gets hard to understand and also may cause incorrect results due to the lack of maintenance. Moreover, the presence of conditional logic on tests does not make it clear which structure of the production code is being covered. Understanding which practices are most prevalent in the professionals' activities supports improving test quality. Other problems are related to incomplete, outdated, or lack of documentation, which make difficult to reach traceability, evolution, and maintenance testing tasks.

6   Threats to validity

Internal validity. Although there are more than 100 test smells, this study only considered 14 of them. However, we selected the most frequently smells discussed in the literature. In addition, the smells were presented in the survey as practices. To mitigate ambiguities and text comprehension, we applied a pilot with 4 testers from different companies.
External validity. We sent the survey to 305 professionals, but only received 60 answers. Although our results may not generalize, they provide an initial view of the practices adopted by testers. Despite the limitations, we performed the survey procedure and data was made available to allow further replications of this study.
Construct validity. During the survey, it was not informed that the questions referred to test smells to do not influence the results. For open questions, a peer-reviewed coding process was performed to avoid bias. The survey and answers were written in Portuguese and translated into English by one author, but reviewed by all others.
Conclusion validity. Since this is a qualitative study, we cannot use a statistical argument to generalize the results. We mitigated this threat by sending the questionnaire to different companies and states of Brazil.

7   Related work

Bavota et al. [2] presented a case study to investigate the impact of test smells on maintenance activities. In that study, developers and students analyzed testing code to compare whether their experience would make a difference in identifying test smells. As a result, they identified that test smells have a significantly negative impact on maintenance activities. Conversely, our study identified for the professionals that we surveyed, the experience does not interfere in the test smell introduction during the creation and execution of a test.

Palomba and Zaidman [12] conducted a study to analyze the relationship between test smells and flaky tests. They analyzed test smells identified in a

multivocal review and detected five flakiness-inducing test smell types [5]. After that, a semi-structured interview was conducted with 10 developers with more than 10 years of experience. As a result, no new test smell was found as a flakiness-inducing, and the five previously identified test smells were ratified. In our study, we were careful not to use the expression test smell as a means to prevent any influence on the respondents' answers. Besides, professionals with different backgrounds participated in the survey.

Tufano et al. [15] proposed a study with 19 participants to investigate developers' perception of test smells. They performed an empirical investigation to analyze where test smells occur at source code. The results showed that developers generally do not recognize test smells, and they are usually introduced since the first commit at the repository. Our study found that most of the professionals frequently adopt practices that lead to smells. However, we use the term *practices* instead of *test smells*. We did not find any study investigating how professional practices affect the introduction of test smells.

8   Concluding Remarks

Test smells may decrease the quality and maintenance of the testing code. Our study aimed at identifying whether professionals know that they introduce smells during test activities, besides understanding whether professional experience influences the adoption of test smells. Our initial results showed that they commit bad practices out of habit or because they follow company standards. We also found that no bad practice was utterly unknown, and all 14 are adopted, even if rarely.

Furthermore, we found that experienced professionals do not insert a few smells than inexperienced ones. As future work, we intend to extend this study to understand better how the industry deals with test smells. We also intend to investigate which techniques companies and professionals could adopt to reduce vices that may lead to test smells.

References

1. Bavota, G., Qusef, A., Oliveto, R., Lucia, A., Binkley, D.: An empirical analysis of the distribution of unit test smells and their impact on software maintenance. In: 28th IEEE International Conference on Software Maintenance (ICSM) (2012)
2. Bavota, G., Qusef, A., Oliveto, R., Lucia, A., Binkley, D.: Are test smells really harmful? An empirical study. Empirical Software Engineering **20**(4) (2015)
3. Fraser, G., Arcuri, A.: Evosuite: Automatic test suite generation for object-oriented software. In: 13th European Conference on Foundations of Software Engineering. ESEC/FSE, ACM, New York, NY, USA (2011)

4. Garousi, V., Felderer, M.: Developing, verifying, and maintaining high-quality automated test scripts. IEEE Software **33**(3) (2016)
5. Garousi, V., Küçük, B.: Smells in software test code: A survey of knowledge in industry and academia. Journal of systems and software **138** (2018)
6. Greiler, M., van Deursen, A., Storey, M.: Automated detection of test fixture strategies and smells. In: 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation (2013)
7. Kitchenham, B.A., Pfleeger, S.L.: Principles of survey research part 2: designing a survey. ACM SIGSOFT Software Engineering Notes **27**(1) (2002)
8. Kitchenham, B.A., Budgen, D., Brereton, P.: Evidence-based software engineering and systematic reviews, vol. 4. CRC press (2015)
9. Linåker, J., Sulaman, S.M., Maiani de Mello, R., Höst, M.: Guidelines for conducting surveys in software engineering. Tech. rep., [Publisher information missing] (2015), `https://lup.lub.lu.se/search/ws/files/6062997/5463412.pdf`
10. Meszaros, G., Smith, S.M., Andrea, J.: The test automation manifesto. In: Maurer, F., Wells, D. (eds.) Extreme Programming and Agile Methods - XP/Agile Universe 2003. Springer Berlin Heidelberg (2003)
11. Palomba, F., Di Nucci, D., Panichella, A., Oliveto, R., De Lucia, A.: On the diffusion of test smells in automatically generated test code: An empirical study. In: 9th International Workshop on Search-based Software Testing. ACM (2016)
12. Palomba, F., Zaidman, A.: The smell of fear: On the relation between test smells and flaky tests. Empirical Software Engineering (2019)
13. Peruma, A.S.A.: What the Smell? An Empirical Investigation on the Distribution and Severity of Test Smells in Open Source Android Applications. PhD Thesis, Rochester Institute of Technology (2018)
14. Smeets, N., Simons, A.J.: Automated unit testing with Randoop, JWalk and μJava versus manual JUnit testing. Research report, Department of Computer Science, University of Sheffield/University of Antwerp, Sheffield, Antwerp (2011)
15. Tufano, M., Palomba, F., Bavota, G., Di Penta, M., Oliveto, R., De Lucia, A., Poshyvanyk, D.: An empirical investigation into the nature of test smells. In: 31st International Conference on Automated Software Engineering. IEEE (2016)
16. Van Deursen, A., Moonen, L., Van Den Bergh, A., Kok, G.: Refactoring test code. In: Proceedings of the 2nd international conference on extreme programming and flexible processes in software engineering (XP) (2001)
17. Van Rompaey, B., Du Bois, B., Demeyer, S.: Characterizing the relative significance of a test smell. In: 22nd International Conference on Software Maintenance (2006)
18. Virgínio, T., Santana, R., Martins, L.A., Soares, L.R., Costa, H., Machado, I.: On the influence of test smells on test coverage. In: Proceedings of the XXXIII Brazilian Symposium on Software Engineering. ACM (2019)
19. Wiederseiner, C., Jolly, S.A., Garousi, V., Eskandar, M.M.: An open-source tool for automated generation of black-box xunit test code and its industrial evaluation. In: Bottaci, L., Fraser, G. (eds.) Testing – Practice and Research Techniques. Springer Berlin Heidelberg (2010)
20. Yusifoğlu, V.G., Amannejad, Y., Can, A.B.: Software test-code engineering: A systematic mapping. Information and Software Technology **58** (2015)