Beyond Green Tests: Removing Smells From Natural Language Tests

Manoel Aranda III
Federal University of Alagoas
Maceió, Brazil
mpat@ic.ufal.br

Márcio Ribeiro
Federal University of Alagoas
Maceió, Brazil
marcio@ic.ufal.br

ABSTRACT

Test smells signal design flaws in tests, harming maintainability and reliability. While automated test smells are well-studied, natural language test smells remain underexplored. Prior work identified 13 such smells but lacked systematic removal strategies and automated tools. We bridge this gap by presenting a catalog of transformations for seven key natural language test smells and a NLP-based tool for automated detection and correction. We evaluated our approach through a survey of 15 professionals and empirical analysis of Ubuntu OS test cases. Results show high professional acceptance (91.43%) and strong tool precision (83.70% F-Measure). Our work is the first to systematically address natural language test smell removal.

KEYWORDS

Natural Language Test, Test Smells, Software Testing

1 Introduction

Test smells indicate poor design decisions in tests, causing maintainability issues, non-deterministic scenarios, and missing verifications [14]. While automated test smells are well-studied with extensive catalogs and tools [1, 4, 6, 12], natural language test smells remain underexplored. Previous work identified 13 natural language test smells [7, 11] but lacked catalogs of transformations to remove the smells and tools to automate the process.

We address this gap by introducing: (1) a catalog of transformations to remove seven natural language test smells: *Unverified Action*, *Misplaced Precondition*, *Misplaced Action*, *Misplaced Verification*, *Eager Action*, *Ambiguous Test*, and *Conditional Test*; and (2) an NLP-based tool that automatically identifies and removes these smells sequentially.

To evaluate our work, we conduct a two-fold empirical strategy. First, we survey 15 software testing professionals from a smartphone manufacturer to answer **RQ**$_1$: *"How do software testing professionals perceive our transformations?"* Results show an average acceptance of 91.43%. Second, we evaluate our tool's precision using Ubuntu OS natural language tests to answer **RQ**$_2$: *"How precise is our tool in removing natural language test smells?"* After analyzing 973 tests and identifying 8,386 smell occurrences, we used Cochran's Sample Size Formula [3] to analyze 264 randomly selected occurrences, achieving 83.70% F-Measure rate.

This paper is a compressed version of our Distinguished Paper Award winner from 2024 [2].

2 Natural Language Test Smells

Previous works showed that test smells also exist in natural language tests [7, 11]. We now present two examples of test smells in a natural language test. We also discuss why it could be harmful during testing activities. All examples of natural language tests in this paper are extracted from the Ubuntu OS manual test repository [13].

A smell is called *Eager Action*, which happens when a single step groups multiple actions. Figure 1 depicts an example of the *Eager Action* smell. This smell introduce problems such as non-isolated tests and difficulties in debugging. In our example, we have four actions, represented by the verbs "*select*", "*enter*", "*select*", and "*click*." Since we have one verification for four actions, in the event of a failure, it may be unclear which action caused it.

Test name: firefox/firefox-006		
Preconditions: This test will check that Firefox can print websites.		
#	**Actions**	**Verifications**
...	...	...
3	Select "print to file" as printer and enter "firefox.pdf" as filename. Select your home folder as location. Then click on "Print"	A window opens, showing the progress of the print
...	...	...

Figure 1: Eager Action - Example

3 Removing Natural Language Test Smells

This section introduces a catalog of transformations to remove test smells in natural language tests. Regarding our terminology, according to [5], refactorings preserve the observable behavior of a code. On the other hand, [14] define test refactorings as changes in test code that do not add or remove test cases. Since our proposals do not strictly align with Fowler's and van Deursen's definitions because they may alter the test behavior and even introduce new test cases, we call them *transformations*.

3.1 Natural Language Test Template

To better explain our transformations, we first need to introduce a template to represent natural language tests. We define a natural language test as $T = (P, S_1, S_2, S_3, \ldots, S_i, \ldots, S_n)$. Where P are preconditions, and S are steps. Each step has actions and verifications. The complete description is on the full paper. Figure 2 presents T and all its elements in terms of a table.

T		
Preconditions: P		
#	Actions	Verifications
S_1	$[a_{11}, a_{12}, \ldots, a_{1n}]$	$[v_{11}, v_{12}, \ldots, v_{1n}]$
...	...	...
S_i	A_i	V_i
...	...	...
S_n	$[a_{n1}, a_{n2}, \ldots, a_{nn}]$	$[v_{n1}, v_{n2}, \ldots, v_{nn}]$

Figure 2: Natural Language Test Template

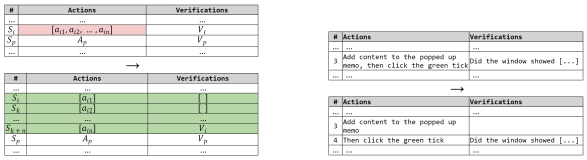
3.2 A Catalog of Transformations

Our catalog considers transformations with left-hand side (with smell) and right-hand side (without smell) [12]. Each transformation includes: (i) addressed smell, (ii) transformation mechanics, (iii) implications (when transformations remove one smell but add another), and (iv) example. We propose transformations for seven natural language test smells: *Unverified Action*, *Misplaced Precondition*, *Misplaced Action*, *Misplaced Verification*, *Eager Action*, *Ambiguous Test*, and *Conditional Test* [7, 11]. Due to space restrictions, we present only one transformation - Fill Verification, which addresses the *Unverified Action* smell. The complete catalog is available in our paper [2].

3.2.1 Separate Actions. Formalization. Figure 3a illustrates that A_i contains n elements, characterizing the *Eager Action* smell: $|A_i| > 1$. To remove the smell, we define a new step for each element of the A_i list. Additionally, in our transformation, we associate the verifications list V_i with the step created for the last action originally in A_i, *i.e.*, the a_{in} action. We opt for this because we consider that the tester will check the verifications list V_i only after executing the last action.

Implications. The *Separate Actions* transformation may lead to the *Unverified Action* smell. After the transformation, we have empty verifications lists for the Steps from S_i to S_{k+n-1}. To deal with the *Unverified Action* smell, we introduce the *Fill Verification* transformation.

Example. Figure 3b shows Step 3 with two actions: "*Add content to the popped up memo*" and "*Then click the green tick*." According to our transformation, they should be split in two different steps. Moreover, the verification "*Did the window showed [...]*" has been associated with the last action, *i.e.*, "*Then click the green tick*."



(a) Transformation **(b) Example**

Figure 3: Separate Actions

4 Catalog Evaluation

This section presents our evaluation, conducted as an online survey with software testing professionals. Here we aim to answer the research question **RQ$_1$: "How do software testing professionals perceive and evaluate the transformations of our catalog?"**

4.1 Planning

We evaluated transformations through an online survey with testing professionals from a smartphone manufacturer using Ubuntu OS test snippets. Each question provided: test smell definition, problem description, identification steps, original/transformed samples, and asked *"Do you agree that, in the example below, the identified problem was addressed according to the definition?"* Responses used a Likert scale from *"I strongly agree"* to *"I strongly disagree"* with optional comments. Participants were recruited via email.

4.2 Results

The survey achieved 15 responses. Demographics: 71% work in industry (vs. academia) with 4.1 years average testing experience. Geographic distribution: 14 participants from Brazil, 1 from Portugal. Figure 4 shows participants' opinions on transformation samples.

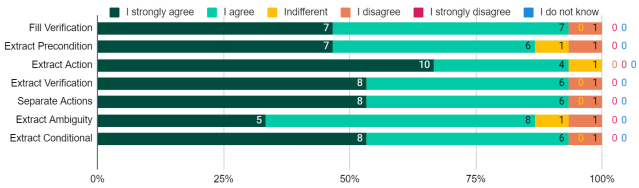


Figure 4: Online Survey Results

4.3 Discussion

Our transformations achieved high validation with 91.43% average acceptance. *Extract Action* received unanimous support (93.3% acceptance), while *Fill Verification*, *Separate Actions*, *Extract Verification*, and *Extract Conditional* achieved 93.3% rates. ***Answer to RQ$_1$.*** The online survey shows software testing professionals mostly agree with our proposals.

5 A Tool to Remove Smells from Manual Tests

For the tool, we use spaCy [8], an NLP library for POS tagging [9], dependency parsing [10], and NER. It uses linguistic patterns to identify test smells. We identify **imperative verbs** using `POS=VERB`, `VerbForm=Inf`, and `Dependency=ROOT` properties, supporting *Separate Actions*.

Our tool considers a sequence we defined based on the implications of each transformation of our catalog.

6 Tool Evaluation

6.1 Planning

We evaluated our tool's precision in removing natural language test smells using 973 Ubuntu OS tests, yielding 8,386 smell occurrences. Due to manual validation constraints, we analyzed a statistically representative sample of 264 occurrences (90% confidence, 5% margin of error using Cochran's formula [3]). The authors validated each transformation, reaching consensus in all disagreement cases (8 out of 264).

We collected the results in terms of true positives (TP)—the smell was corrected—, false positives (FP)—the smell was not corrected—, and false negatives (FN)—no correction was attempted.

6.2 Results

Our tool achieved 86.75% Precision, 80.85% Recall, and 83.70% F-Measure. The *Fill Verification* transformation performed best (89.55% TP), while *Separate Actions* had the lowest success rate (50.72% TP,

Table 1: Total and sample occurrences per test smell

Test Smell	Total	Sample
Unverified Action	1,967	67
Misplaced Precondition	49	5
Misplaced Action	345	8
Misplaced Verification	426	16
Eager Action	2,663	69
Ambiguous Test	2,656	90
Conditional Test	279	9
TOTAL	8, 386	264

36.23% FN). Two transformations (*Extract Action* and *Extract Verification*) showed 25% false positive rates, primarily due to spaCy's handling of special characters and compound words like *"Re-Check."*

Table 2: TP, FP, and FN per transformation

Transformation	TP %	FP %	FN %
Fill Verification	89.55%	2.99%	7.46%
Extract Precondition	80.00%	0.00%	20.00%
Extract Action	62.50%	25.00%	12.50%
Extract Verification	62.50%	25.00%	12.50%
Separate Actions	50.72%	13.04%	36.23%
Extract Ambiguity	76.67%	12.22%	11.11%
Extract Conditional	77.78%	11.11%	11.11%

6.3 Discussion

Poor test writing (wrong phrases, excessive special characters, wrong formatting) causes malfunctions, adding special characters and breaking phrases. Special characters also led *Extract Action* and *Extract Verification* to apply incorrectly.

Answer to RQ$_2$. Our analysis shows promising tool results, achieving a F-Measure of 83.70%.

6.4 Threats to Validity

We found **external validity** concerns (evaluation limited to tests from Ubuntu OS) and **internal validity** risks from manual analysis subjectivity, mitigated through authors consensus validation.

7 Related Work

Natural language test smells were first introduced by [7], who proposed seven smells and keyword-based identification techniques. Ten years later, [11] extended this work with six additional smells using NLP-based detection methods. However, neither work provided transformations or automated tools for smell removal, which our work addresses.

8 Concluding Remarks

We introduced a catalog of transformations to remove natural language test smells. We assessed the quality of our catalog by recruiting 15 software testing professionals. The professionals found the transformations valuable, which is supported by the high average acceptance rate. We also introduced a tool that implements the catalog. We executed the tool in 264 occurrences of test smells and achieved 83.70% of F-Measure rate. As future work, we intend to (i) increase the set of transformations to include other smells; and (ii) use Large Language Models (LLMs) to improve the effectiveness of our tool.

REFERENCES

[1] Diogo Almeida, José Creissac Campos, João Saraiva, and João Carlos Silva. 2015. Towards a catalog of usability smells. In *SAC 2015*. 175–181.
[2] Manoel Aranda, Naelson Oliveira, Elvys Soares, Márcio Ribeiro, Davi Romão, Ullyanne Patriota, Rohit Gheyi, Emerson Souza, and Ivan Machado. 2024. A Catalog of Transformations to Remove Smells From Natural Language Tests. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering* (Salerno, Italy) *(EASE '24)*. Association for Computing Machinery, New York, NY, USA, 7–16. doi:10.1145/3661167.3661225
[3] James E. Bartlett II, Joe W. Kotrlik, and Chadwick C. Higgins. 2001. Organizational research: Determining appropriate sample size in survey research. *Information technology, learning, and performance journal* 19, 1 (2001), 43–50.
[4] 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 (2015), 1052–1094.
[5] Martin Fowler and Kent Beck. 1997. Refactoring: Improving the design of existing code.
[6] Vahid Garousi and Barış Küçük. 2018. Smells in software test code: A survey of knowledge in industry and academia. *Journal of systems and software* 138 (2018), 52–81.
[7] Benedikt Hauptmann, Maximilian Junker, Sebastian Eder, Lars Heinemann, Rudolf Vaas, and Peter Braun. 2013. Hunting for smells in natural language tests. In *ICSE 2013*. 1217–1220.
[8] Matthew Honnibal and Ines Montani. 2024. *spaCy – Industrial-strength Natural Language Processing in Python*. https://spacy.io/
[9] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19, 2 (1993), 313–330. https://aclanthology.org/J93-2004
[10] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *LREC 16*. 1659–1666.
[11] Elvys Soares, Manoel Aranda, Naelson Oliveira, Márcio Ribeiro, Rohit Gheyi, Emerson Souza, Ivan Machado, André Santos, Baldoino Fonseca, and Rodrigo Bonifácio. 2023. Manual tests do smell! cataloging and identifying natural language test smells. In *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–11.
[12] Elvys Soares, Márcio Ribeiro, Rohit Gheyi, Guilherme Amaral, and André Santos. 2023. Refactoring Test Smells With JUnit 5: Why Should Developers Keep Up-to-Date? *IEEE Transactions on Software Engineering* 49, 3 (2023), 1152–1170.
[13] Ubuntu. 2024. *Ubuntu Manual Tests in Launchpad*. https://launchpad.net/ubuntu-manual-tests
[14] Arie Van Deursen, Leon Moonen, Alex Van Den Bergh, and Gerard Kok. 2001. Refactoring test code. In *XP 2001*. 92–95.