Manual Tests Do Smell! Cataloging and Identifying Natural Language Test Smells

Elvys Soares*, Manoel Aranda†, Naelson Oliveira†, Márcio Ribeiro†, Rohit Gheyi‡, Emerson Souza*,
Ivan Machado§, André Santos*, Baldoino Fonseca†, and Rodrigo Bonifácio¶
*Universidade Federal de Pernambuco (UFPE), Brazil
Email: eas5@cin.ufpe.br, epss@cin.ufpe.br, alms@cin.ufpe.br
†Federal University of Alagoas (UFAL), Brazil
Email: mpat@ic.ufal.br, naelson@ic.ufal.br, marcio@ic.ufal.br, baldoino@ic.ufal.br
‡Federal University of Campina Grande (UFCG), Brazil
Email: rohit@dsc.ufcg.edu.br
§Federal University of Bahia (UFBA), Brazil
Email: ivan.machado@ufba.br
¶University of Brasília (UnB), Brazil
Email: rbonifacio@unb.br

***Abstract*—Background: Test smells indicate potential problems in the design and implementation of automated software tests that may negatively impact test code maintainability, coverage, and reliability. When poorly described, manual tests written in natural language may suffer from related problems, which enable their analysis from the point of view of test smells. Despite the possible prejudice to manually tested software products, little is known about test smells in manual tests, which results in many open questions regarding their types, frequency, and harm to tests written in natural language. Aims: Therefore, this study aims to contribute to a catalog of test smells for manual tests. Method: We perform a two-fold empirical strategy. First, an exploratory study in manual tests of three systems: the Ubuntu Operational System, the Brazilian Electronic Voting Machine, and the User Interface of a large smartphone manufacturer. We use our findings to propose a catalog of eight test smells and identification rules based on syntactical and morphological text analysis, validating our catalog with 24 in-company test engineers. Second, using our proposals, we create a tool based on Natural Language Processing (NLP) to analyze the subject systems' tests, validating the results. Results: We observed the occurrence of eight test smells. A survey of 24 in-company test professionals showed that 80.7% agreed with our catalog definitions and examples. Our NLP-based tool achieved a precision of 92%, recall of 95%, and f-measure of 93.5%, and its execution evidenced 13,169 occurrences of our cataloged test smells in the analyzed systems. Conclusion: We contribute with a catalog of natural language test smells and novel detection strategies that better explore the capabilities of current NLP mechanisms with promising results and reduced effort to analyze tests written in different idioms.**

***Index Terms*—Test Design, Software/Program Verification, Test Smells, Manual Tests, Natural Language Processing**

I. Introduction

Test smells are indications of potential problems in the design and implementation of automated software tests [1]. Like a code smell [2], [3], a test smell does not necessarily mean an already existing problem but an indication of further difficulties such as poor maintainability (*i.e.,* duplication of code [1]), lack of coverage (*i.e.,* missing or unexecuted verifications [4]), or unreliable results (*i.e.,* non-deterministic execution behavior [5]).

The necessary investment in configuration can lead a project to opt for manual testing over test automation due to budget limitations [6], [7]. In such cases, manual test descriptions are in natural language and *"often of poor quality and written without the best practices of software engineering"* [8]. Similar to known issues with natural language requirements, documentation of tests in natural language often results in test cases that are incomprehensible, ambiguous, and difficult to maintain due to problems such as translation and spelling errors, different description styles for similar testing procedures, or excessive use of abbreviations [9].

Despite the format differences, bad choices when implementing automatic tests [10] or describing a manual test using natural language may pose similar threats to the testing activity. For example, Table I presents a fragment of a test description from the Ubuntu Operational System (OS) manual tests.[1] In the test, the second action step presents two conditions, *"USB 3.0 storage device"* and *"USB 3.0 port,"* that must be met for the action *"transfer a large file"* and the corresponding verification step to be performed. The conditional logic phrased in natural language negatively affects test comprehension and correctness. Indeed, as can be seen in Table I, a problem in USB 3.0 file transfers may not be identified if the tester does not use compliant equipment and skip step 2. From the point of view of test smells, this is the Conditional Test [11] in natural language [6].

Using the rationale presented by the example in Table I, Hauptmann *et al.* [6] coined the term *natural language test smells* to represent possible design problems in manual software testing from the point of view of test smells. A set of seven test smells is presented along with simple detection

[1][Online]. *"testcases\hardware\1476_USB Ports"* test, available: https://git.launchpad.net/ubuntu-manual-tests

978-1-6654-5223-6/23/$31.00

TABLE I
STEPS OF AN UBUNTU OS TEST HAVING THE CONDITIONAL TEST SMELL PHRASED IN NATURAL LANGUAGE

No	Action	Verification
1	Plug a USB device in and attempt to use it	The device is correctly recognized. The software normally used with the device functions normally. The device behaves as expected. The USB device works in every port. You are able to disconnect and re-connect the USB device correctly without errors
2	*If the device is a USB 3.0 storage device and you have a USB 3.0 port, transfer a large file between the two*	*The transfer is above USB 2.0 speed*
3	Repeat for each USB device you have	

rules, such as word count or occurrences from keyword lists. After Hauptmann *et al.* [6] publication, we noticed a research gap of almost ten years concerning natural language test smells, which did not happen in the context of smells in automatic tests [5], [12]–[14]. Such absence motivates a handful of questions concerning the existence of additional natural language test smells, their frequency, the possible problems they indicate, and whether we can benefit from the powerful Natural Language Processing mechanisms available nowadays. These questions motivate our work in this paper, which aims to advance the research on natural language test smells.

We first conduct an exploratory study to analyze a statistically relevant sample of manual test descriptions of three systems from different domains: (i) the Ubuntu Operational System (OS), which is open-source; (ii) the Brazilian Electronic Voting Machine, in an institutional partnership between our institution [name omitted for the blind review process] and the Superior Electoral Court (TSE); and (iii) a large smartphone manufacturer's UI — name omitted due to non-disclosure of proprietary information agreement —, also in partnership. In this first study, we intend to answer the following research questions:

- **RQ$_1$:***"What already proposed natural language test smells can be observed?"*,
- **RQ$_2$:***"What new natural language test smells can be observed?"*, and
- **RQ$_3$:***"How frequent are these test smells?"*

Answering these research questions is important to advance the list of test smells applicable to natural language tests. In particular, we identify the occurrence of two already proposed natural language test smells (*i.e., Conditional Test* and *Ambiguous Test*) and contribute to six new smells (*i.e., Unverified Action*, *Misplaced Precondition*, *Misplaced Verification*, *Misplaced Action*, *Eager Action*, and *Tacit Knowledge*), and their frequency in the systems mentioned above. As the final product of this study, we introduce a catalog containing these eight smells. Our catalog organizes each smell in terms of their name, definition, problem, and identification rules. Instead of using simple detection mechanisms (*e.g.,* searching for a keyword to identify a test smell), our rules are based on powerful natural language processing capabilities like the identification of indefinite determiners, which may indicate non-determinism in the test description.

We conduct an empirical study using an online survey to evaluate our catalog. We recruited 24 test professionals and presented them with our definitions and examples, asking for their agreement level to our propositions. In this study, we intend to answer the following research question:

- **RQ$_4$:***"How software testing professionals evaluate our proposed smells?"*

As a result of our survey, our proposals had an average acceptance of 80.7% among the interviewed in-company test engineers, contributing to additional concerns, such as test reproducibility, length, maintainability, and coverage, all originating from doubts raised from poor test writing.

We also contribute to developing an NLP-based tool to identify our catalog's natural language test smells automatically. Our tool implements our defined rules using spaCy,[2] a *"free, open-source library for industrial-strength Natural Language Processing (NLP) in Python,"*, and its capabilities concerning syntactic analysis (i.e., elements of the sentences and their properties) like verification verbs and declarative sentences, which are present in multiple languages and whose implementation can be is mostly reused — as we do to Portuguese, used in the tests of the Brazilian electronic voting machine. To evaluate our tool, we conduct one last empirical study to answer the following research question:

- **RQ$_5$:***"How precise can the automated discovery of natural language test smells be when using NLP?"*

The results of this study point to a precision of 92%, recall of 95%, and f-measure of 93.5%, indicating a suitable detection level for our proposals. Overall, the tool execution evidenced 13,169 test smell occurrences in the 2,007 tests of the analyzed systems, which, by definition, may represent enhancement opportunities to their descriptions.

To sum up, this paper presents the following contributions:

- We conduct an exploratory study for natural language test smells on systems of different domains: open-source, government, and industry (Section II);
- We present a catalog of natural language test smells, with six new contributions from our study, along with detection rules that use syntactic and morphological language analysis, representing a novel approach enabled by current NLP technology (Section III);
- We evaluate our catalog with 24 in-company test engineers (Section IV);

[2][Online]. Available: https://spacy.io/

- We introduce a NLP-based tool to identify the proposed test smells (Section V);
- We evaluate our tool by analyzing a sample of its results concerning the before-mentioned systems (Section VI).

The survey dataset, tool logs, and tool validation records — for Ubuntu OS tests — are available online [15].

II. Exploratory Study: Towards a Catalog of Natural Language Test Smells

This section describes how we analyzed natural language test descriptions to prospect test smells. Also, we give further detail on the selected systems, a sample set of tests, and the distribution (frequency) of our findings. In particular, this exploratory study answers $\mathbf{RQ}_1$, $\mathbf{RQ}_2$, and $\mathbf{RQ}_3$.

A. Planning

This exploratory study aims to prospect a set of manual tests from different systems and gather the identified occurrences of test smells. To increase the representativeness of our results, we selected manual tests written in natural language from important systems of three distinct domains: open-source, government, and industry. Considering the limits imposed by the agreements for non-disclosure of confidential information, we detail the obtained tests as follows:

Ubuntu OS: As open-source software [16], the Ubuntu OS manual tests are available in a public repository.[3] In the repository, test descriptions are in English and XML format, with standardized tags for test suites, test cases, and action and verification steps. In total, 305 test files containing 973 tests are available.

Brazilian Electronic Voting Machine (BEVM): An open-source web-based test management and test execution system manages the manual test descriptions of the BEVM. In the ecosystem, test descriptions are in Portuguese. In total, we had access to 133 tests exported to HTML format.

Large Smartphone Manufacturer (LSM): The manual test descriptions of this industry partner are managed by a proprietary issue-tracking product that allows bug tracking and agile project management. Manual test descriptions for this system are in English. In total, 898 test descriptions were made available for our analysis and exported to spreadsheet format.

Three authors manually and independently analyzed a randomly selected subset of test descriptions to perform the exploratory study. Using their know-how on test smells for automatic and manual tests, the authors quoted every questionable description and indicated the possible smell, discussing results in follow-up meetings. It is important to emphasize that access to BEVM and LSM tests was controlled and accessed by cleared authors only. As to the analysis procedure, all authors involved in this activity started with the Ubuntu manual tests to achieve standardization of actions, continuing the analysis in the remaining systems according to their access grants.

[3][Online]. Available: https://git.launchpad.net/ubuntu-manual-tests

Concerning the already proposed smells for tests written in natural language, from the existing list of seven test smells [6], five are identified using metrics from an automatic analysis [17]: *Badly Structured Test Suite*, *Inconsistent Wording*, *Hard-Coded values*, *Long Test Steps*, and *Test Clones*. As we intended to manually read test descriptions and take notes of the identified problems, using any tool to generate such metrics was out of scope.

Finally, to make our manual analysis effort feasible, we used Cochran's Sample Size Formula [18] to calculate the sample needed to obtain an 80% confidence level with a 5% margin of error for each system individually. Table II presents the analyzed sample test set per system:

TABLE II
Analyzed sample set of tests per system.

System	Manual tests	Sample size
Ubuntu OS	973	141
BEVM	136	75
LSM	898	139
Total	**2,007**	**355**

B. Results

We found similarities in all systems regarding the structure of their manual tests. Although Ubuntu's team does not use a specific test managing tool, they describe their tests as the other two systems, which use open-source and proprietary software for such activities. Fig. 1 presents a test visualization. Table III details the test section's writing, regarding the sentence types, with examples from the Ubuntu OS tests.

Test name		
Objective		
Preconditions		
Steps:		
1	action	verification
...	...	...
n	action	verification

Fig. 1. Common test design found in the exploratory study.

TABLE III
Common test structure found in the exploratory study.

Section	Sentence type	Example
Objective	Declarative	*This test checks that Audio project menu Works*
Preconditions	Declarative	*VMWare Player version $\geq$ 4.0 is required*
	Imperative	*Ensure that your system has no Internet access before proceeding*
Action	Imperative	*Click the 'Restart now' button*
Verification	Declarative	*An 'Installation Complete' dialog appears*
	Imperative	*Verify the system upgraded correctly*

The exploratory study identified eight test smells, briefly defined in Table IV and further detailed in Section III. From this list, two smells (*i.e., Ambiguous Test* and *Conditional*

Test) are proposals from the literature on natural language test smells [6], and the remaining ones are contributions from our study. Also, we manually accounted for 447 occurrences of the identified test smells, and Fig. 2 presents their distribution per system.

TABLE IV
CATALOGED TEST SMELLS

Test Smell	Brief definition
Ambiguous Test	Test steps leaving room for interpretation
Conditional Test	Conditional logic phrased in natural language
Eager Action	Single action steps that group multiple actions
Misplaced Action	Action steps written as verification steps
Misplaced Precondition	Preconditions as action steps
Misplaced Verification	Verification steps written as action steps
Tacit Knowledge	Unexplained terms and abbreviations
Unverified Action	Action steps without corresponding verifications

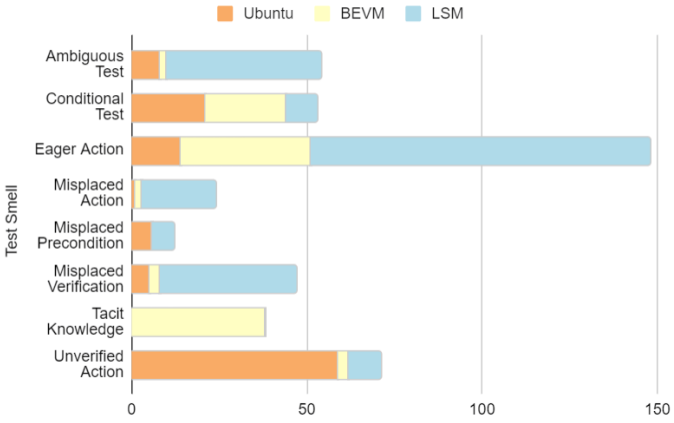

Fig. 2. Distribution of identified test smells per system.

C. Discussion

The structural test pattern found in the analyzed systems (Section II-B) enabled us to propose the test smells presented in Section III. Moreover, the distribution of such smells demonstrates the analysis of natural language test descriptions from the point of view of test smells to present promising results. The manual analysis offers some insights whose reality will be precisely shown in Section V. These insights, for now, indicate that:

- Most observed test smells are common to all analyzed systems (*e.g., Eager Action*);
- There are test smells unique to a single system (*e.g., Tacit Knowledge*);
- Each system has its own test smell trend (*e.g.,* Ubuntu tests suffer more from *Unverified Action*).

Summary: Answering $\mathbf{RQ}_1$, we could observe two already proposed natural language test smells in the analyzed systems. In addition, six new test smells are observed, which answers $\mathbf{RQ}_2$. Answering $\mathbf{RQ}_3$, the test smells are frequent throughout the analyzed systems. In particular, Eager Action and Tacit Knowledge tend to be the most and the least frequent ones.

D. Threats to Validity

a) Conclusion: Our identified test smells relate to the common test structure in all three analyzed systems. However, it is important to notice that BEVM and LSM tests are managed by well-adopted software solutions throughout the industry, leading us to understand the found pattern as generally widespread, possibly minimizing this threat.

b) Internal: As the accuracy of the exploratory study (*i.e.,* 80%) is not ideal for generalizations in the analyzed systems (*i.e.,* 95% [18]), the distribution of test smells presented in Fig. 2 may not be precise. We minimized this problem by modeling and validating a Natural Language Processing (NLP)-based tool, further detailed in Section V, to provide the exact distribution of the presented test smells.

c) External: As external threats, analyzing a few software systems may not be enough to identify relevant or well-spread test smells. We minimize this probability by using systems representative of different domains and spoken languages and finding test smells common to such systems.

III. A CATALOG OF NATURAL LANGUAGE TEST SMELLS

We now present our catalog, the main product of our exploratory study. We show the identified test smells in terms of their names, definition, problem, and identification rules for their detection with examples from the analyzed Ubuntu OS test descriptions.

A. Ambiguous Test

a) Definition: Originally proposed by Hauptmann *et al.* [6], this smell indicates an *"under-specified test that leaves room for interpretation"*.

b) Problem: It negatively impacts test comprehension and execution, since the aim needs to be clarified and multiple test executions are not comparable [6].

c) Identification: The original detection rule was the occurrence of any word from a fixed list of "vague words."[4] As Hauptmann *et al.*'s [6] keyword list originated from occurrences in their analyzed test suites and we found a slightly different list in our exploratory study (*e.g.,* some, other, and any), we noticed such keywords to be common in their semantics (syntactic analysis). We propose a more general set of detection rules which consider keyword semantics, and examples, in Table V.

B. Conditional Test

a) Definition: Tests containing conditional logic phrased in natural language.

b) Problem: The Conditional Test turns tests very complex and difficult to maintain, negatively impacting test comprehension and correctness since it is hard to understand the intention, and complex tests are more likely to have errors [6].

[4]similar, better, similarly, worse, having in mind, take into account, take into consideration, clear, easy, strong, good, bad, efficient, useful, significant, adequate, fast, recent, far, close

TABLE V
AMBIGUOUS TEST IDENTIFICATION

Rule	Example
Verb + indefinite determiner	***Open any*** *application and suspend machine*
Indefinite pronouns	*At "Write changes to disks", verify that* ***everything*** *is right and select YES*
Comparative adjectives	*Is the performance* ***similar*** *or* ***better*** *with no graphical display issues?*
Superlative adjectives	*The root filesystem uses* ***most*** *of the SD card.*
Adverbs of manner	*Does fast user switching work* ***quickly****?*
Comparative adverbs	*Does everything function* ***better*** *than the stable version?*

c) Identification: Originally, Hauptmann *et al.* [6] proposed a fixed list of words for its detection.[5] As the list is non-exhaustive concerning subordinating conjunctions, we propose any subordinating conjunction, as in Table VI, to identify this smell as a more robust detection rule.

TABLE VI
CONDITIONAL TEST IDENTIFICATION

Rule	Example
Subordinating conjunctions	***If*** *you have a USB drive, plug it in.*

C. Eager Action

a) Definition: Single action steps that group multiple actions.

b) Problem: This test smell may hide implementation problems when any action lacks verification, negatively affecting test effectiveness.

c) Identification: Imperative verbs represent actions. Example in Table VII.

TABLE VII
EAGER ACTION IDENTIFICATION

Rule	Example
Multiple imperative verbs	***Change*** *some sound settings or other settings (night mode, call history, SMS, etc.) and* ***display*** *them on the phone,* ***download*** *some applications, etc.*

D. Misplaced Action

a) Definition: Indicative of a structurally malformed test, the Misplaced Action smell arises when action steps are written as results.

b) Problem: It negatively impacts test maintainability, since the test structure is not consistent.

c) Identification: Imperative verbs, excluding verification verbs,[6] present in verification steps. Example in Table VIII.

E. Misplaced Precondition

a) Definition: Also an indicative of structurally malformed tests, here, preconditions are written as action steps.

[5]if, whether, depending, when, in case

[6]Verification verbs identified in use: check, verify, observe, recheck

TABLE VIII
MISPLACED ACTION IDENTIFICATION

Rule	Example
Imperatives, excluding verification verbs, as verification steps	***Give*** *a name to the directory and* ***add*** *files to it as you* ***did*** *in the previous step*

b) Problem: Difficulties in test correctness, since the incorrect placement of preconditions may influence the tester to report test failure should a precondition be unattended.

c) Identification: When the first action step declares the SUT state. The common format of SUT state is a *noun (subject)* followed by an *auxiliary verb*, followed by a *past participle* verb or adjective in the same sentence (Table IX).

TABLE IX
MISPLACED PRECONDITION IDENTIFICATION

Rule	Example
Subject followed by an auxiliary verb followed by another verb on the past participle	*The* ***monitor is*** *not* ***connected****, and the* ***PC is*** *not* ***paired***

F. Misplaced Verification

a) Definition: Another indicative of structurally malformed tests, this smell arises when verification steps written as action steps.

b) Problem: It negatively impacts test maintainability, since the test structure is not consistent.

c) Identification: Sentences containing verification verbs written as or along with action steps. Example in Table X.

TABLE X
MISPLACED VERIFICATION IDENTIFICATION

Rule	Example
Verification in or as an action step	*Close flip and* ***check*** *app continuity*

G. Tacit Knowledge

a) Definition: This test smell is related to the use of unexplained terms and abbreviations presuming the tester's familiarity to domain-specific definitions.

b) Problem: It negatively impacts test comprehension and execution.

c) Identification: Abbreviations and domain-specific terms not explained in the test description or external reference document (*i.e.,* glossary). A hypothetical example, since we are not authorized to disclose BEVM tests, is in Table XI.

TABLE XI
TACIT KNOWLEDGE IDENTIFICATION

Rule	Example
Unexplained terms and abbreviations	*Check for reported* ***residual votes***

H. Unverified Action

a) Definition: Action steps that miss corresponding verification steps.

b) Problem: Absent verification steps negatively affect test execution and correctness since there is no instruction on how the system should behave, leaving room for the testers' interpretation.

c) Identification: Action steps with no corresponding verification steps.

IV. Catalog Evaluation

In this section, we present the online survey performed to evaluate our proposals. This activity, in particular, answers $\mathbf{RQ}_4$.

A. Planning

This study planned to assess the opinions of software testing professionals (*e.g.*, engineers, analysts, and managers) about the manual test smells we proposed in Section III through an online survey. By stating their agreement with our definitions and examples and commenting on their answers, the software testing professionals would validate whether our proposals represented valid test smells in theory and practice.

We assembled an online survey with questions corresponding to the given definition and example, same as presented in Section III. Respondents were presented with the following answering options (unique): *"I strongly agree"*, *"I agree"*, *"Indifferent"*, *"I disagree"*, and *"I strongly disagree"*. Also, every question had an optional comment field.

We recruited participants through individual email invitations, using emails from our industry partner. The invitations were sent to participants of manual test teams, quality assurance professionals, and test managers, none of whom had compensation or obligation to respond to the survey.

B. Results

We performed the survey in March 2023, achieving 24 responses for 110 sent emails (21.8% response rate). Concerning the demographics, we had participants from Brazilian teams, where 83.3% defined their primary work area as the industry — over academia — and their average declared experience with software testing was 4.3 years. Fig. 3 details the results concerning the participants' opinions on our proposals.

C. Discussion

Regarding the proposed test smells (Section III), the opinion of experienced test professionals (industry partner) served as validation that obtained a high acceptance rate (Fig. 3). We present the details in the following paragraphs.

Already present in the literature, the *Ambiguous Test* smell (Section III-A) definition was ratified by 83% of the respondents. Among the agreeing comments, the ambiguity may indeed cause tests to be poorly performed depending on the tester experience, as in *"My experience can improve the test coverage, however for a beginner tester is not be clear the ways to test an interruption, and this can induce he/she to repeat the same procedure/routine or try few different ways to suspend the app."* Among the testers that disagree with the test smell definition, the variance allowed by non-deterministic terms is beneficial to test different scenarios, as seen in *"I would say that exploratory test cases use a similar approach and it has been working"*.

Known in automatic and manual testing, the *Conditional Test* (Section III-B) smell definition and example had the acceptance of 83% of the respondents. Concerns about the conditionals being able to improve the test coverage on features not always available arise in both sides, as in the agreeing opinion *"The only part I would not agree is if it is related to a feature that the product may not actually have implemented, for example, NFC."*, and the disagreeing opinion *"The test writer attempted to cover more possible verifications. If a step or accessory can't be verified all the test is not blocked, and the test becomes applicable to different kinds of product"*.

As the first proposal of our work, the *Eager Action* (Section III-C) test smell definition was ratified by 87.5% of respondents, with no disagreeing opinion. Among the comments, difficulties in the test execution and concerns about the verifications can be found in *"it seems rather confusing and not pointing to any settings overall, it is covering multiple scenarios"* and in *"There isn't a guarantee that the tester checked all configurations available"*.

Ratified by 75% of the respondents, the *Misplaced Action* (Section III-D) test smell had supporters that manifested concerns about test structure, as in *"Verification steps should be in the end of test cases. Preconditions at the beginning, and actions in the middle."* Testers that do not agree with the given definition manifest no concern to test structure, since they comprehend the test objective, as in *"If the action keeps the step valid as a single one, it makes sense to be written"*.

The concerns described by the *Misplaced Precondition* (Section III-E) test smell definition were accepted by 87.5% of the respondents. Unfortunately, as that was not a mandatory task in this survey, the respondents provided no comments to this test smell description.

The *Misplaced Verification* (Section III-F) test smell was our least accepted proposal, even though counting with 62.5% agreeing opinions. Testers claim the test clarity to benefit from the separation into action and verification steps, as in *"I agree because I think that is more clear and organized for the test have it in separated (verification) steps"*. On the opposite hand, testers that did not agree also claimed maintainability benefits of keeping action and verification steps written together, as in *"these actions help to avoid too many steps in a script and reduces the effort in test maintenance"*.

Our most accepted proposal, the *Tacit Knowledge* test smell (Section III-G) definition had the support of 91.6% respondents. The excessive use of abbreviations and unexplained domain-specific terms is indeed a concern to agreeing respondents, as in *"In my experience, I have faced many new testers and interns having problems knowing abbreviations in test cases"*. Disagreements call attention to test maintainability, as in *"I would say it is a case by case scenario where it*

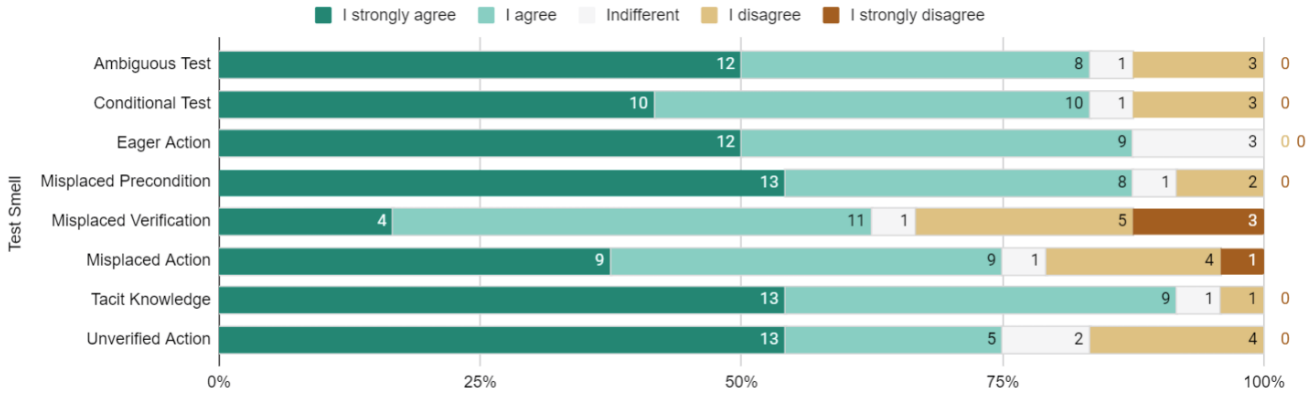

Fig. 3. Survey results

could be bad either way. I could have overly long texts due to unnecessary repetition that could be solved by Basic Glossary before the TCs (test cases). Or a inverse scenario where the tester is not provided with edge information to that test."

Our last proposal, the *Unverified Action* test smell (Section III-H) had the approval of 75% respondents. No agreeing respondent gave further details on their answer. Disagreeing respondents manifested concern about the verification steps to every action, as in *"Not every action, in a sequence of actions, generates a relevant result to be verified."* and *"In some situations the expected result is too obvious and can be dispensed. I believe that this helps to not tire the reader."*

Summary: The online survey shows software testing professionals to mostly agree with our proposals. In addition to positively answering $\mathbf{RQ}_4$, their comments show additional concerns, such as test reproducibility, length, maintainability, and coverage, all originated from poor test writing.

D. Threats to Validity

Concerning the internal results, some respondents made the same claim for better organization when a test has action and verification steps written together or separated, for instance, both representing agreeing and disagreeing opinions. However, the wide acceptance of our proposals votes in favor of our interpretation of the possible prejudices, minimizing the threat.

We used responses from software testing professionals who work for our industrial partner, and this bias may influence the generalization of results to other audiences. We minimize this probability through the respondents' experience, of about 4.3 years in average (Section IV-B), and whose answers tend to be similar to experienced professionals who test software in other domains.

Concerning the construct validity, the lack of an attention checking question could bias the results towards the confirmation of our proposals. We minimize this threat by providing examples with attention terms stressed with bold fonts, as in Section III examples.

V. A Tool to Detect Smells in Manual Tests

We present the development of an NLP-based tool, which we call Manual Test Sensei, to detect the natural language test smells we described in Section III. This effort shows how implementing our rules for natural language test smells identification is feasible using the current state of the NLP technology.

We use Python and spaCy [19], a commercial open-source software released under the MIT license [20], to implement the NLP tool containing our rules for discovering natural language test smells. SpaCy features convolutional neural network models for part-of-speech (POS) tagging [21], dependency parsing [22], text categorization, and named entity recognition (NER) [23]. Fig. 4 shows a visualization of the dependency parsing — arrows above the sentence — and the POS tagging — labels beneath each sentence element — for the Conditional Test example of Section III-B.

The motivation for choosing this combination of programming language and NLP library were (i) using market tools focused on results and performance to analyze industrial-scale software and (ii) the availability of language models beyond English since BEVM tests are in Portuguese.

The chosen strategy enabled us to implement most of our identification proposals. However, identifying the *Tacit Knowledge* (Section III-G) requires a more comprehensive solution. To perform it, one would consider (i) external documentation (*e.g.,* glossaries and execution manuals) — non-existent in Ubuntu and not provided in the BEVM and LSM — and (ii) a list of standard terms used in manual software testing and considered tacit in every manual testing scenario, where every outsider term would characterize the *Tacit Knowledge* test smell if not clarified. To our knowledge, the proposition of such a list requires a formal study.

Also, we had to consider the different test file formats according to each analyzed project (Section II-A): XML for the Ubuntu OS, HTML for the BEVM tests, and spreadsheet for LSM tests. To that end, specific parsers were created for each system's test file format. Fig. 5 presents a simplified UML class diagram of the Manual Test Sensei tool, where the parsers — responsible for transforming a test file into several test objects — and the test smell matchers are shown. Finally, the tool produces a CSV file as output containing the test file name, the identified test smell, the specific words or sentence

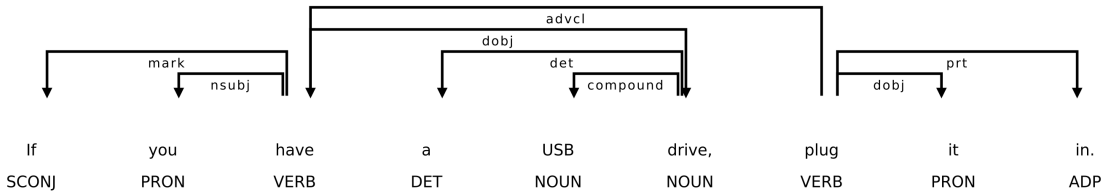

Fig. 4. spaCy's visualizer module example

span that characterize the test smell, and the analyzed (action or verification) step.

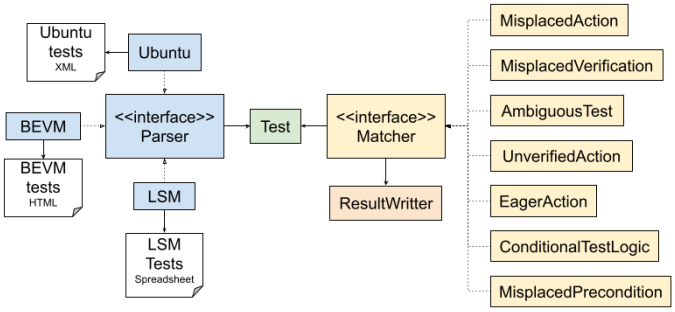

Fig. 5. Simplified UML class diagram of the developed NLP tool

The tool source code is available in an online repository at https://github.com/easy-software-ufal/manual-test-sensei.

VI. Tool Evaluation

Once the proposition and development of the Manual Test Sensei tool — implementing our natural language test smell identification rules — proved possible using current NLP technology (Section V), in this last study, we present the tool results and validation, therefore demonstrating how precise is the tool performance. This activity, in particular, answers $\mathbf{RQ}_5$.

A. Planning

This study planned to execute the Manual Test Sensei tool against the entire test set of the three analyzed systems and validate the results. Therefore, we could verify whether the distribution found in the exploratory study (Section II) is maintained in the Manual Test Sensei execution results, as well as the accuracy — in terms of precision, recall, and f-measure metrics [24], [25] — of such results.

Although we executed our tool against the entire test set of the three systems, manually validating the tool's output of 13,169 smells would be infeasible. Therefore, we randomly selected 101 tests distributed in proportion to the number of tests available in every analyzed system.

For every selected test, an author would first analyze it manually and indicate the found test smells, then verify the tool results for that test, and finally indicate the results that were correct or true positives (TP), incorrect or false positives (FP), and the missed or false negatives (FN) test smells. Table XII presents the distribution of the randomly selected tests per system:

TABLE XII
Distribution of selected tests in the validation sample

System	Total tests	Sample size
Ubuntu OS	973	49
BEVM	136	7
LSM	898	45
Total	**2,007**	**101**

B. Results

A total of 2,007 test descriptions were analyzed by the Manual Test Sensei tool. The tool indicated 13,169 test smells, with an average of 6.5 test smells per analyzed test, noticeably higher than the 1.2 test smells found in the exploratory study (Section II). Considering the analyzed systems individually, we obtained an average of 8.5 test smells per Ubuntu OS test, 5.8 test smells per BEVM test, and 4.5 test smells per LSM test. Table XIII presents the results per test smell and system. Finally, a distribution of the found test smells per analyzed system is presented in Fig. 6.

TABLE XIII
Total NLP results

Test Smell	Ubuntu	BEVM	LSM	Total
Ambiguous Test	2,627	185	1,776	**4,588**
Conditional Test	277	110	193	**580**
Eager Action	2,664	299	1,191	**4,154**
Misplaced Action	318	19	124	**461**
Misplaced Precondition	45	3	74	**122**
Misplaced Verification	428	161	513	**1,102**
Unverified Action	1,967	11	184	**2,162**
Total	**8,326**	**788**	**4,055**	**13,169**

Three authors performed the verification as defined in Section VI-A. The selected sample of 101 tests resulted in 708 results for this activity. The analysis of such results by the involved authors resulted in 15 disagreements comprising doubts in syntactical and morphological text analysis, properly clarified in a discussion meeting. Table XIV presents the detailed validation totals per system and the precision, recall, and f-measure metrics achieved by the tool in this validation activity.

C. Discussion

The high expressiveness of the adopted technology, either in the identification of dependency relationships (e.g., subject + auxiliary verb + participle verb) or in the identification of

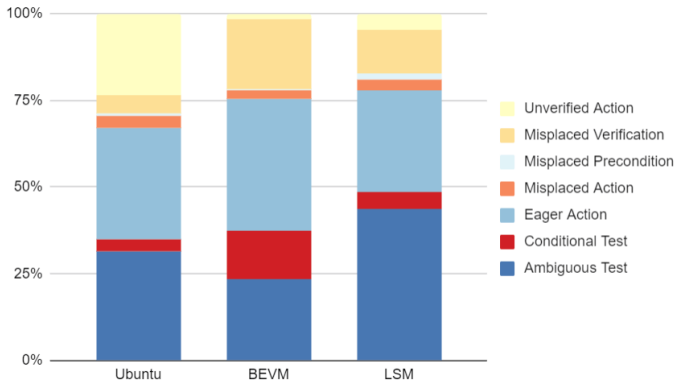

Fig. 6. Distribution of test smells per system.

TABLE XIV
DETAILED NLP TOOL VALIDATION AND METRICS

System	TP	FP	FN	Precision	Recall	F-measure
Ubuntu OS	384	43	18	0.9	0.96	92.64
BEVM	25	0	0	1	1	1
LSM	213	13	12	0.94	0.95	94.46
Total	**622**	**56**	**30**	**0.92**	**0.95**	**93.53**

the Part of Speech (POS) (e.g., indefinite pronouns), enabled us to implement most of the detection rules as defined in Section III. Only one identification rule could not be implemented entirely, which was the Conditional Test, identified through subordinating conjunctions (SCONJ) at the beginning of a dependent clause in a sentence. As spaCy does not natively support splitting sentences into clauses, which varies from language to language, identifying SCONJ in a dependent clause in the middle of a sentence results in many identification problems by the pre-trained models. This problem resulted in 8 false negatives identified in the validation activity, representing approximately 27% of the test smells not identified by the tool.

We encountered various formatting, spelling, and character encoding conversion issues in the test descriptions. Using numbered and unnumbered lists, parentheses, and the lack of correct punctuation impaired the NLP engine classification in some cases reported as false positives and false negatives. For example, the implemented mechanism was not able to identify a subordinate clause in the sentence *"(If on a 'laptop') Is plugged to a power source,"* nor in *"Type in your user name and press Enter (you can accept the default if you wish),"* and could not differentiate the link label in the sentence *"Click the Choose Payment Method link,"* which lacked quotes, and was erroneously classified as multiple actions.

However, even with the implementation challenges and some test malformations mentioned, the result obtained in the metrics of precision, recall, and f-measure for the tool can still be considered expressive. The results remain promising even when using a trained model for a different idiom and executing the same rules — except for the list of verification verbs used in the *Misplaced Verification* detection, which needed a partner in Portuguese for BEVM tests — as seen in the metrics presented by Table XIV.

According to Table XIII, the most frequent test smells detected were the *Ambiguous Test* (*i.e.,* 34.8%) and *Eager Action* (*i.e.,* 31.5%). An interesting distribution noticed is that, from the 4,588 occurrences of the *Ambiguous Test*, we accounted for 2,225 (*i.e.,* 48.5%) occurring in action steps and 2,363 (*i.e.,* 51.5%) occurrences in verification steps, meaning that ambiguous tests have an almost equal probability of presenting testers with difficulties in *"what to perform in the test"* and *"what to verify as a result."* However, being less frequent may not mean less harm to the testing activity. It is important to remember that a *Misplaced Precondition* can induce the tester to declare the test failed if the precondition is not met and the test cannot be executed [26], [27].

Comparing the distribution of test smells found in the exploratory study (Section II) and the one found by the NLP tool (Section VI-B), shown in Fig. 7, we noticed that some test smells had a different percentage result between the two activities, which was the case of the *Ambiguous Test* and the *Conditional Test*. This expressive difference was due to the more precise identification of the tool in cases of undefined determinants, which may escape the most attentive — or not sufficiently trained in the exploratory study — eyes. Still, the precision difference in the exploratory study (Section II) and the tool validation (Section VI-B), necessary for this study to be feasible, influenced the found deviation.

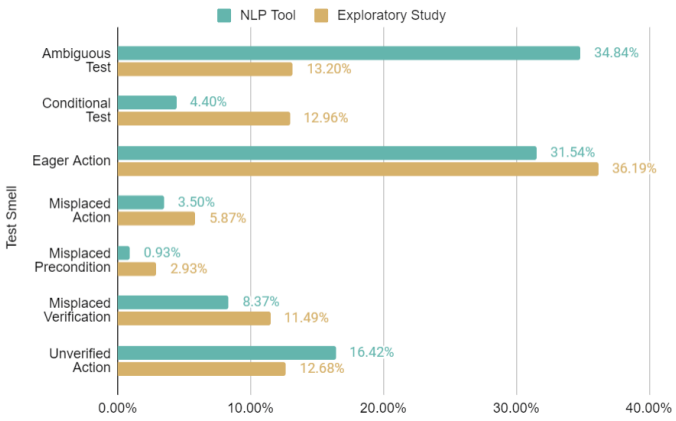

Fig. 7. Comparison between the exploratory study and the NLP tool results.

Furthermore, we noticed that test smells not found in the exploratory study for specific systems, such as the *Misplaced Precondition* for the BEVM tests, are now among the results of the NLP tool (Table XIII), even with few occurrences (i.e., 3). This result is also expected and included in the exploratory study's 5% margin of error (Section II-A). Finally, the proportional distribution of test smells per system shown in Fig. 6 shows that although the tests of the analyzed systems suffer from the test smells found, they do so in different proportions.

Summary: The results obtained in the tool validation show that our detection rules are effective in identifying the considered test smells. In particular, we achieved a f-measure of

93.53%, which answers $\mathbf{RQ}_5$.

D. Threats to Validity

Internal: The tool's results may contain errors. We manually analyzed 101 tests to minimize this threat, which meant more than 700 results, according to Table XIV. This amount of results was enough to guarantee statistical validity [18] for the 13,169 results generated by the tool.

External: The generalization of the obtained results is impossible with the selected sample of three systems. We minimize this threat by choosing highly expressive systems from different domains to analyze. Nevertheless, an exploratory study would confirm whether our results indicate some degree of probability to the analysis of other systems.

VII. Related Work

Hauptmann *et al.* [6] presented possible problems in manual test descriptions performed in natural language from the point of view of test smells. Together with coining the term *Natural Language Test Smells*, the authors propose a set of seven smells: *Hard-Coded Values*, *Long Test Steps*, *Conditional Tests*, *Badly Structured Test Suites*, *Test Clones*, *Ambiguous Tests*, and *Inconsistent Wording*. Also, the authors present identification strategies for their proposals that rely on keyword lists and complimentary metrics (*i.e., number of words*) and the frequency of the proposed test smells in nine industrial test suites. In our work, we extend the current catalog by adding six new test smells, their discovery strategies and frequency, and providing updates for the discovery of two of Hauptmann *et al.*'s list, which we base on broader definitions focused on morphological and syntactical language analysis, thus exploring the capabilities of current Natural Language Processing mechanisms.

Rajkovic and Enoiu presented a tool called NALABS to detect bad smells in natural language requirements and test specifications [28]. Similarly to Hauptmann *et al.* [6], the proposed tool uses keyword lists to measure vagueness, referenceability, optionality, subjectivity, and weakness metrics. They also used Automated Readability Index (ARI) to measure readability and the number of words and conjunctions to measure test complexity. Again, our work differentiates from Rajkovic and Enoiu's work because we use current NLP mechanisms to identify words using morphological and syntactical language analysis.

Transferring the concept of code smells to requirements engineering, Femmer *et al.* [29] introduced a lightweight static requirements analysis approach that allows for quick checks when requirements are written down in natural language. In another work, Femmer *et al.* [30] derived a set of smells from the natural language criteria of the ISO/IEC/IEEE 29148 standard, showing that lightweight smell analysis can uncover many practically relevant requirements defects. Like our work, they also use tool support to analyze text in natural language descriptions.

Previous works presented test smells in test code. Some of these smells are related to ours, although we focused on natural language test smells. Meszaros *et al.* [11] and Peruma *et al.* [13] studied test smells in test code, such as *Conditional Test* and *Conditional Test Logic*, which are related to Hauptmann *et al.* [6] natural language test smell. Aljedaani *et al.* [31] also listed the *Assertionless Test* smell, defined by the absence of assertions, which is similar to our idea of natural language tests having no verification steps (*Unverified Action*).

VIII. Concluding Remarks and Future Work

In this paper, we extended the current research on Natural Language Test Smells by contributing six new test smell propositions, strategies for their detection, and their frequency in a sample of three representative systems in the government, open-source, and industry domains. Also, we proposed updates for two well-known test smells applicable to natural language test descriptions. Unlike the current research, we proposed a novel detection strategy for natural language test smells that relies on syntactical and morphological text analysis, thus exploring the capabilities of current Natural Language Processing mechanisms.

To conduct this work, we performed two independent and complimentary parts: first, we performed an exploratory study whose results we validated with industry test professionals. We guided the development of an NLP-based tool in the second part. The results and validation metrics showed our strategy to be effective, detecting test smells with over 90% precision, even in a multiple-idiom context.

In future work, we intend to (i) enable the implementation of the *Tacit Knowledge* test smell by performing a formal study to define common terms in software testing terminology that may be considered tacit in any manual execution of software tests; (ii) execute the tool analysis in other candidate systems whose test management is performed using the same tools as BEVM and LSM tests to verify the generalization of our results; and (iii) aggregate tests — and test file formats — from uncovered systems in the results.

Finally, we verified that some cataloged test smells also exist in the case of automatic tests. Moreover, the results of this study show that, like their automatic correlates, natural language test smells are also quite frequent, corroborating the title statement.

Acknowledgment

We thank the Brazilian Superior Electoral Court (TSE) and our industrial partner for kindly allowing their tests to be analyzed in our study. This research was partially funded by CNPq grants 312195/2021-4, 421306/2018-1, 310313/2022-8; and FAPEAL grants 60030.0000000462/2020 and 60030.0000000161/2022. Also, this work is partially supported by INES (National Institute of Software Engineering): CNPq grant 465614/2014-0, CAPES grant 88887.136410/2017-00, and FACEPE grants APQ-0399-1.03/17 and PRONEX APQ/0388-1.03/14.

REFERENCES

[1] A. van Deursen, L. Moonen, A. van Den Bergh, and G. Kok, "Refactoring test code," in *2nd International Conference on eXtreme Programming and Flexible Processes in Software Engineering*, ser. XP. USA: CiteSeer, 2001, pp. 92–95.

[2] N. Oliveira, M. Ribeiro, R. Bonifácio, R. Gheyi, I. Wiese, and B. Fonseca, "Lint-based warnings in python code: Frequency, awareness and refactoring," in *2022 IEEE 22nd International Working Conference on Source Code Analysis and Manipulation (SCAM)*, ser. SCAM 2022, 2022, pp. 208–218.

[3] F. Medeiros, G. Lima, G. Amaral, S. Apel, C. Kästner, M. Ribeiro, and R. Gheyi, "An investigation of misunderstanding code patterns in c open-source software projects," *Empirical Software Engineering*, vol. 24, pp. 1693–1726, 2019.

[4] A. Tahir, S. Counsell, and S. G. MacDonell, "An empirical study into the relationship between class features and test smells," in *APSEC*. New York, NY, USA: IEEE, 2016, pp. 137–144.

[5] F. Palomba and A. Zaidman, "The smell of fear: On the relation between test smells and flaky tests," *Empirical Software Engineering*, vol. 24, no. 5, pp. 2907–2946, 2019.

[6] B. Hauptmann, M. Junker, S. Eder, L. Heinemann, R. Vaas, and P. Braun, "Hunting for smells in natural language tests," in *35th International Conference on Software Engineering*, ser. ICSE. New York, NY, USA: IEEE, 2013, pp. 1217–1220.

[7] L. Fernandes, M. Ribeiro, R. Gheyi, M. Delamaro, M. Guimarães, and A. Santos, "Put your hands in the air! reducing manual effort in mutation testing," in *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*, ser. SBES '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 198–207.

[8] B. Hauptmann, "Reducing system testing effort by focusing on commonalities in test procedures," Ph.D. dissertation, Technische Universität München, Germany, Jul 2016.

[9] K. Juhnke, A. Nikic, and M. Tichy, "Clustering natural language test case instructions as input for deriving automotive testing dsls," *Journal of Object Technology*, vol. 20, no. 3, pp. 1–14, 2021.

[10] F. Dalton, M. Ribeiro, G. Pinto, L. Fernandes, R. Gheyi, and B. Fonseca, "Is exceptional behavior testing an exception? an empirical assessment using java automated tests," in *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 170–179.

[11] G. Meszaros, *xUnit test patterns: Refactoring test code*. Boston, USA: Pearson Education, 2007.

[12] V. Garousi and B. Küçük, "Smells in software test code: A survey of knowledge in industry and academia," *Journal of Systems and Software*, vol. 138, pp. 52–81, 2018.

[13] A. Peruma, K. Almalki, C. D. Newman, M. W. Mkaouer, A. Ouni, and F. Palomba, "On the distribution of test smells in open source android applications: An exploratory study," in *29th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON. USA: IBM Corp, 2019, pp. 193–202.

[14] A. Panichella, S. Panichella, G. Fraser, A. A. Sawant, and V. J. Hellendoorn, "Test smells 20 years later: detectability, validity, and reliability," *Empirical Software Engineering*, vol. 27, no. 7, p. 170, 2022.

[15] E. Soares, M. Terceiro, N. Oliveira, M. Ribeiro, R. Gheyi, E. Souza, I. Machado, A. Santos, B. Fonseca, and R. Bonifácio, "Manual Tests Do Smell! Cataloging and Identifying Natural Language Test Smell - Replication Package," 7 2023. [Online]. Available: http://doi.org/10.6084/m9.figshare.22652620.v2

[16] C. Ltd., "Ubuntu operational system," https://ubuntu.com/download, [Accessed 02-May-2023].

[17] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[18] J. E. Bartlett II, J. W. Kotrlik, and C. C. Higgins, "Organizational research: Determining appropriate sample size in survey research," *Information technology, learning, and performance journal*, vol. 19, no. 1, pp. 43–50, 2001.

[19] M. Honnibal and I. Montani. spacy – industrial-strength natural language processing in python. [Online]. Available: https://spacy.io/

[20] J. H. Saltzer, "The origin of the "mit license"," *IEEE Annals of the History of Computing*, vol. 42, no. 4, pp. 94–98, 2020.

[21] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993. [Online]. Available: https://aclanthology.org/J93-2004

[22] J. Nivre, M.-C. de Marneffe, F. Ginter, Y. Goldberg, J. Hajič, C. D. Manning, R. McDonald, S. Petrov, S. Pyysalo, N. Silveira, R. Tsarfaty, and D. Zeman, "Universal Dependencies v1: A multilingual treebank collection," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, May 2016, pp. 1659–1666.

[23] M. Honnibal and I. Montani, "spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To appear*, vol. 7, no. 1, pp. 411–420, 2017.

[24] C. J. Van Rijsbergen, "Foundation of evaluation," *Journal of documentation*, vol. 30, no. 4, pp. 365–373, 1974.

[25] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *arXiv preprint arXiv:2010.16061*, 2020.

[26] E. Soares, M. Ribeiro, G. Amaral, R. Gheyi, L. Fernandes, A. Garcia, B. Fonseca, and A. Santos, "Refactoring test smells: A perspective from open-source developers," in *Proceedings of the 5th Brazilian Symposium on Systematic and Automated Software Testing*, ser. SAST 20. New York, NY, USA: Association for Computing Machinery, 2020, p. 50–59.

[27] E. Soares, M. Ribeiro, R. Gheyi, G. Amaral, and A. Santos, "Refactoring test smells with JUnit 5: Why should developers keep up-to-date?" *IEEE Transactions on Software Engineering*, vol. 49, no. 3, pp. 1152–1170, 2023.

[28] K. Rajkovic and E. P. Enoiu, "Nalabs: Detecting bad smells in natural language requirements and test specifications," Mälardalen Real-Time Research Centre, Mälardalen University, Tech. Rep., February 2022. [Online]. Available: http://www.es.mdu.se/publications/6382-

[29] H. Femmer, D. Méndez Fernández, S. Wagner, and S. Eder, "Rapid quality assurance with requirements smells," *Journal of Systems and Software*, vol. 123, pp. 190–213, 2017.

[30] H. Femmer, D. M. Fernández, E. Juergens, M. Klose, I. Zimmer, and J. Zimmer, "Rapid requirements checks with requirements smells: Two case studies," in *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, ser. RCoSE 2014. New York, NY, USA: Association for Computing Machinery, 2014, pp. 10–19.

[31] W. Aljedaani, A. Peruma, A. Aljohani, M. Alotaibi, M. W. Mkaouer, A. Ouni, C. D. Newman, A. Ghallab, and S. Ludi, "Test smell detection tools: A systematic mapping study," in *Evaluation and Assessment in Software Engineering*, ser. EASE 2021. Association for Computing Machinery, 2021, pp. 170–180.