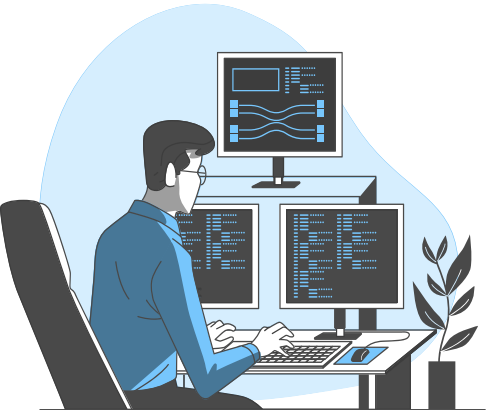
UNT UNIVERSITY OF NORTH TEXAS
RIT
ÉTS Le génie pour l'industrie
25th International Conference on Evaluation and Assessment in Software Engineering (EASE)

Test Smell Detection Tools: A Systematic Mapping Study



Wajdi Aljedaani
University of North Texas

Mazen Alotaibi
Rochester Institute of Technology

Christian D. Newman
Rochester Institute of Technology

Anthony Peruma
Rochester Institute of Technology

Mohamed Wiem Mkaouer
Rochester Institute of Technology

Abdullatif Ghallab
University of North Texas

Ahmed Aljohani
Rochester Institute of Technology

Ali Ouni
ETS Montreal, University of Quebec

Stephanie Ludi
University of North Texas

Context

- Software testing is an essential part of the software development life cycle.
- In 2001 [1], the catalog of test smells has been steadily growing throughout the years.

Ensure

Reliable software

Discovers

Defects/bugs

Guarantees

Quality of the software

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

Problem Statement

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Smells in software test code: A survey of knowledge in industry and academia

Vahid Garousi[a,*], Barış Küçük[b]

[a] Information Technology Group, Wageningen University, Netherlands
[b] Atilim University, Ankara, Turkey

ARTICLE INFO

Article history:
Received 20 April 2017
Revised 9 December 2017
Accepted 13 December 2017
Available online 15 December 2017

Keywords:
Software testing
Automated testing
Test automation
Test scripts
Test smells
Test anti-patterns
Multivocal literature mapping
Survey
Systematic mapping

ABSTRACT

As a type of anti-pattern, test smells are defined as poorly designed tests and their presence may negatively affect the quality of test suites and production code. Test smells are the subject of active discussions among practitioners and researchers, and various guidelines to handle smells are constantly offered for smell prevention, smell detection, and smell correction. Since there is a vast grey literature as well as a large body of research studies in this domain, it is not practical for practitioners and researchers to locate and synthesize such a large literature. Motivated by the above need and to find out what we, as the community, know about smells in test code, we conducted a 'multivocal' literature mapping (classification) on both the scientific literature and also practitioners' grey literature. By surveying all the sources on test smells in both industry (120 sources) and academia (46 sources), 166 sources in total, our review presents the largest catalogue of test smells, along with the summary of guidelines/techniques and the tools to deal with those smells. This article aims to benefit the readers (both practitioners and researchers) by serving as an "index" to the vast body of knowledge in this important area, and by helping them develop high-quality test scripts, and minimize occurrences of test smells and their negative consequences in large test automation projects.

1. Introduction

Software testing can be conducted either manually or in an automated manner. In manual testing, a human tester takes over the role of an end-user interacting with and executing the software under test (SUT)[1] to verify its behavior and to find any observable defects (Amannejad et al., 2014). On the other hand, in automated testing, test-code scripts are developed using certain test tools (e.g., the JUnit framework) and are then executed without human testers' intervention to test the behavior of an SUT. If planned and implemented properly, automated testing could yield various benefits over manual testing, such as repeatability and reduction of test effort (and thus costs). However, if not implemented properly, automated testing will lead to extra costs and effort and could even be less effective than manual testing in detecting faults (Amannejad et al., 2014).

Automated software testing and development of test code (scripts) are now mainstream in the software industry. For instance, in a recent book, Microsoft test engineers reported that *"there were more than a million [automated] test cases written for Microsoft Office 2007"* (Page et al., 2008). Automated test suites with high internal quality facilitate maintenance activities, such as code comprehension and regression testing. Development of test-code scripts is however tedious, error prone and requires significant up-front investment. Furthermore, developing high-quality test-code is not trivial and *"writing effective unit tests is as much about the test itself as it is about the code under test"* (Seguin, 2009). *"Tests can have bugs too!"* (Multiple anonymous authors, 2016). As a test practitioner pointed out in a blog (Seguin, 2009), *"Complex and messy unit tests don't add any value even if the code under test is perfectly designed"*.

Many guidelines have been proposed to help developers develop high-quality test code. We coined the term *'Software Test-Code Engineering (STCE)'* in our recent works (Garousi et al., 2015; Garousi and Felderer, 2016) which refers to the set of practices and methods to systematically develop, verify and maintain high-quality test code. Unfortunately, such practices and guidelines are not always followed properly in practice, resulting in symptoms called *bad smells* (anti-patterns) in test code (or simply *test smells*).

FEATURE: TEST SMELLS

What We Know About Smells in Software Test Code

Vahid Garousi, Wageningen University

Barış Küçük, Proven Information Technologies

Michael Felderer, University of Innsbruck and Blekinge Institute of Technology

// Test smells are poorly designed tests and negatively affect the quality of test suites and production code. We present the largest catalog of test smells, along with a summary of guidelines, techniques, and tools used to deal with test smells. //

SOFTWARE TESTING IS conducted by either manual or automated testing, test-code scripts are find any observable defects. In automated testing, such as repeatability and reduction of test costs (and thus, effort). However, if not properly implemented, automated testing will lead to additional costs and effort and could even be less effective than manual testing in detecting faults.[1]

The automated software testing and development of test code (scripts) are now mainstream in the software industry. For instance, in a recent book, Microsoft test engineers reported that "there were more than a million (automated) test cases written for Microsoft Office 2007."[2] Just like regular source code, automated test suites are also source code and thus should be of highest quality. No one wants to test his/her software under development using a test suite that is of low quality with defects itself. Many guidelines have been proposed to help developers create high-quality automated test suites. We coined the term *software test-code engineering* in our recent works,[1] which refers to the set of practices and methods used to systematically develop, verify, and maintain high-quality test code. Unfortunately, such practices and guidelines are not always followed properly in practice, resulting in symptoms called *bad smells* (antipatterns) in test code (or simply *test smells*). Test smells are defined as poorly designed tests and their presence may negatively affect the maintainability of test suites and production code, or even their functionality.

In their collaborations with practitioner testers and in the context of sev-

Study Goal

The goal is to provide developers and researchers with a one-stop source that can offer comprehensive insight into test smell detection tools.

Research Questions

What test smell detection tools are available to the community, and what are the common smell types they support?

What are the main characteristics of test smell detection tools?

Study Design: Search & Selection

Keywords

Title:("tool*" OR "detect*" OR "test smell" OR "test smells") AND Abstract:("test smell" OR "test smells" OR "test code" OR "unit test smell")

Digital Libraries

- ACM Digital Library
- Scopus
- IEEE Xplore
- Springer Link
- Science Direct
- Web of Science

Study Design: Search & Selection

Inclusion & exclusion search criteria

Inclusion	Exclusion
Published in Computer Science	Websites, leaflets, and grey literature
Written in English	Published in 2021
Available in digital format	Full-text not available online
Propose or use test smell detect tool	Tools not associated with peer-reviewed papers

Table 1: Our inclusion and exclusion search criteria.

Study Design: Search & Selection

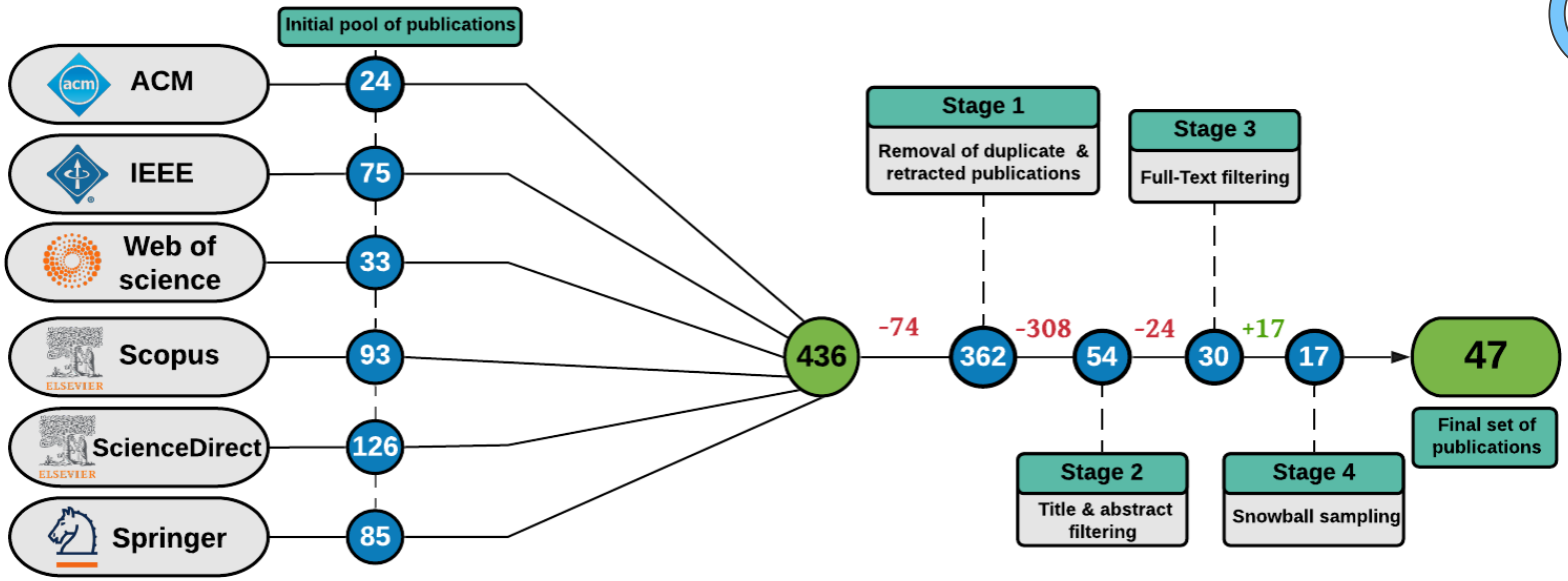


Figure 1: Overview of the volume of publications resulting from our filtering process.

Search Findings

RQ 1

What test smell detection tools are available to the community, and what are the common smell types they support?

Part 1:
Publication Years & Venues

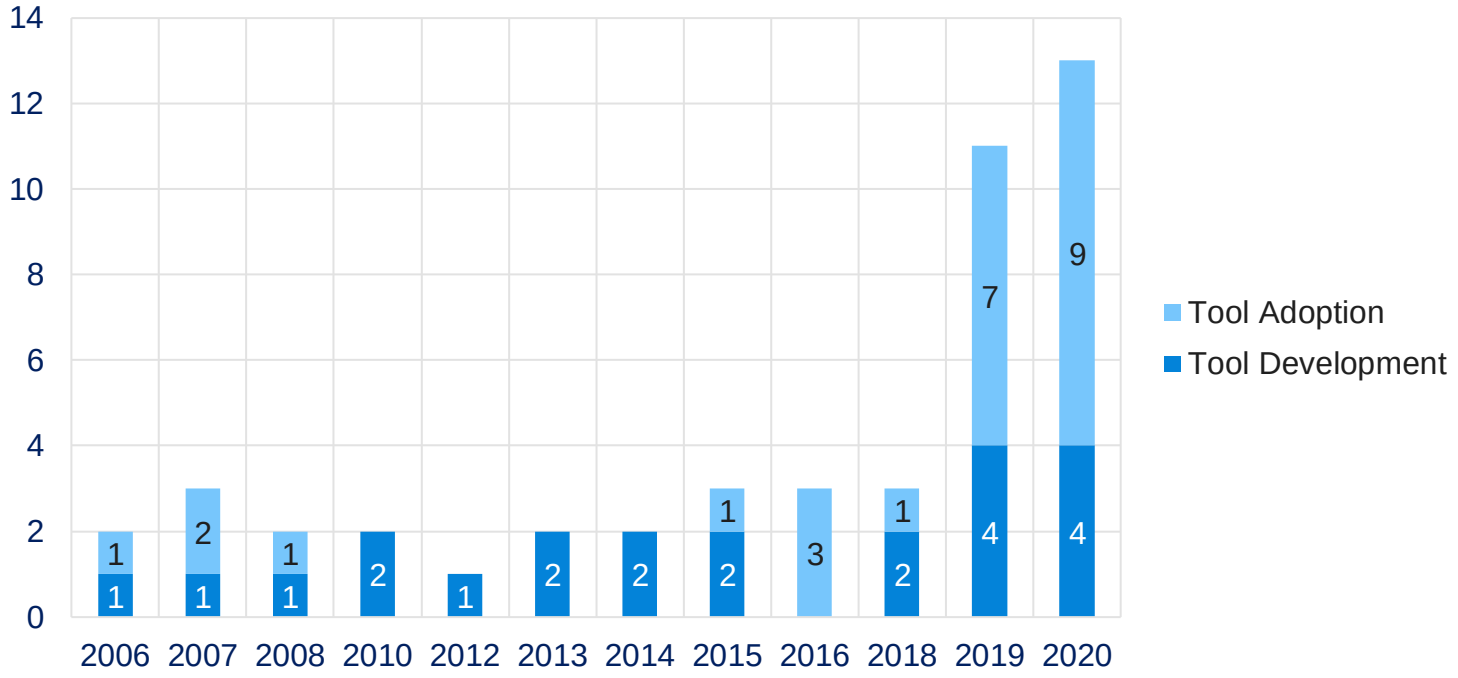


Figure 2: Yearly breakdown of tool publications.

Search Findings

Part 2:
Test Smell Detection Tools

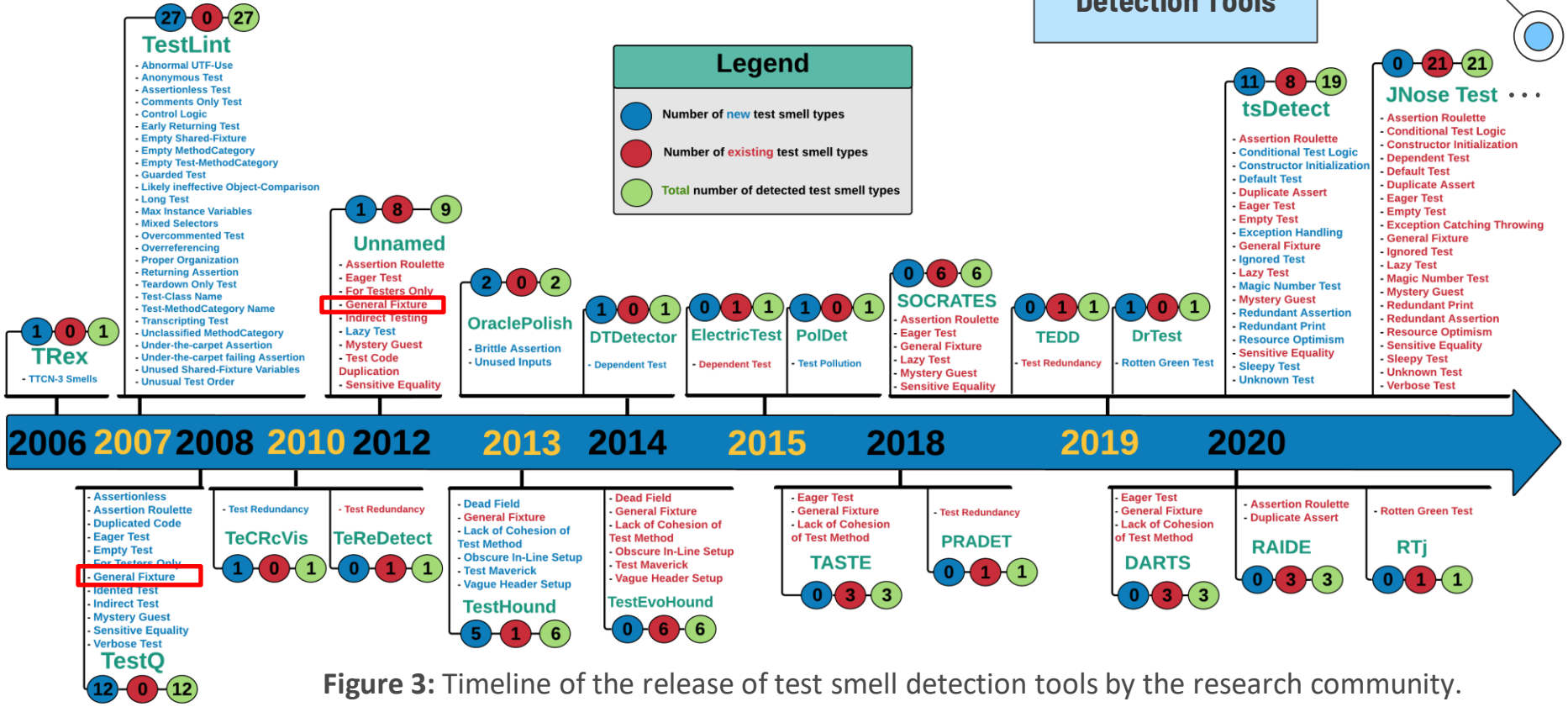


Figure 3: Timeline of the release of test smell detection tools by the research community.

Search Findings

Part 3:
Test Smell Detection Types

Tool \ Smell Type	AL	AR	CI	CTL	DA	DC	DepT	DF	DT	EH	EmT	ET	FTO	GF	IgT	InT	IT	LCM	LT	MG	MNT	OISS	RA	RO	RP	RT	SE	ST	TM	TR	TRW	UT	VHS	VT
DARTS [46]												√		√				√																
DrTest [31]																										√								
DTDetector [75]							√																											
ElectricTest [24]							√																											
JNose Test [72]	√	√	√	√	√		√		√	√	√	√		√					√	√	√		√	√	√		√	√				√		√
PRADET [34]							√																											
RAIDE [60]		√			√																													
RTj [47]																										√								
SoCRATES [30]		√										√		√					√	√							√							
TASTE [52]												√		√				√																
TeCReVis [44]																														√				
TEDD [25]							√																											
TeReDetect [45]																														√				
TestEvoHound [40]								√						√				√				√							√				√	
TestHound [39]								√						√				√				√							√				√	
TestQ [27]	√	√				√					√	√	√	√		√	√			√							√							√
TSDETECT [56]		√	√	√	√				√	√	√	√		√	√				√	√	√		√	√	√		√	√				√		
Unnamed [22]		√				√						√	√	√			√		√	√							√	√			√			
Total	2	6	2	2	3	2	5	2	2	2	3	7	2	9	1	1	2	4	4	5	2	2	2	2	2	2	5	2	2	2	1	2	2	2

Table 2: Distribution of test smells detected by the test smell detection tools.

Search Findings

Part 4:
Supported Programming Languages

Programming Language	Supported Test Smell Types				Literature Usage
Java	(01) Assertion Roulette (AR)	(11) Eager Test (ET)	(21) Magic Number Test (MNT)	(31) Test Maverick (TM)	[20, 49, 50, 73, 74]
	(02) Assertionless (AL)	(12) Empty Test (EmT)	(22) Mystery Guest (MG)	(32) Test Pollution (TP)	[22, 39, 40, 44, 75]
	(03) Brittle Assertion (BA)	(13) Exception Handling (EH)	(23) Obscure In-line Setup Smell (OISS)	(33) Test Redundancy (TR)	[23, 24, 41, 51, 65]
	(04) Conditional Test Logic (CTL)	(14) For Testers Only (FTO)	(24) Redundant Assertion (RA)	(34) Test Run War(TRW)	[34, 55, 58, 63, 72]
	(05) Constructor Initialization (CI)	(15) General Fixture (GF)	(25) Redundant Print (RP)	(35) TTCN-3 Smells (TTCN)	[25, 37, 38, 52, 61]
	(06) Dead Field (DF)	(16) Ignored Test (IgT)	(26) Resource Optimism (RO)	(36) Unknown Test (UT)	[46, 56, 64, 70, 71]
	(07) Default Test (DT)	(17) Indented Test (InT)	(27) Rotten Green Tests (RT)	(37) Unused Input (UI)	[26, 43, 53, 54, 62]
	(08) Dependent Test (DepT)	(18) Indirect Test (IT)	(28) Sensitive Equality (SE)	(38) Vague Header Setup(VHS)	[33, 42, 47, 57, 66]
	(09) Duplicate Assert (DA)	(19) Lack of Cohesion of Test Method (LCM)	(29) Sleepy Test (ST)	(39) Verbose Test (VT)	[45, 60]
	(10) Duplicated Code (DC)	(20) Lazy Test (LT)	(30) (31) Test Code Duplication (TCD)		
Scala	(01) Assertion Roulette (AR)	(03) Exception Handling (EH)	(05) Mystery Guest (MG)		[29, 30]
	(02) Eager Test (ET)	(04) General Fixture (GF)	(06) Sensitive Equality (SE)		
SmallTalk	(01) Abnormal UTF-Use (AUU)	(08) Empty Shared-Fixture (ESF)	(15) Under-the-carpet failing Assertion (UCFA)	(22) Test-Class Name (TCN)	[31, 59]
	(02) Anonymous Test (AT)	(09) Empty Test-MethodCategory (ETMC)	(16) Overcommented Test (OCT)	(23) Test-MethodCategory Name (TMC)	
	(03) Assertionless Test (AL)	(10) Guarded Test (GT)	(17) Overreferencing (OF)	(24) Transcripting Test (TT)	
	(04) Comments Only Test (COT)	(11) Likely ineffective Object-Comparison (LIOC)	(18) Proper Organization (PO)	(25) Unclassied MethodCategory (UMC)	
	(05) Control Logic (ConL)	(13) Long Test (LoT)	(19) Returning Assertion (RA)	(26) Under-the-carpet Assertion (UCA)	
	(06) Early Returning Test (ERT)	(12) Max Instance Variables (MIV)	(20) Rotten Green Tests falls (RT)	(27) Unused Shared-Fixture Variables (USFV)	
	(07) Empty MethodCategory (EMC)	(13) Mixed Selectors (MS)	(21) Teardown Only Test (TOT)	(28) Unusual Test Order (UTO)	
C++	(01) Assertion Roulette (AR)	(04) Eager Test (ET)	(07) General Fixture (GF)	(10) Mystery Guest (MG)	[27]
	(02) Assertionless Test (ALT)	(05) Empty Test (EmT)	(08) Indented Test (InT)	(11) Sensitive Equality (SE)	
	(03) Duplicated Code (DC)	(06) For Testers Only (FTO)	(09) Indirect Test (IT)	(12) Verbose Test (VT)	

Table 3: Distribution of Test Smells Per Programming Languages.

Search Findings

RQ 2 What are the main characteristics of test smell detection tools?

Common Characteristics

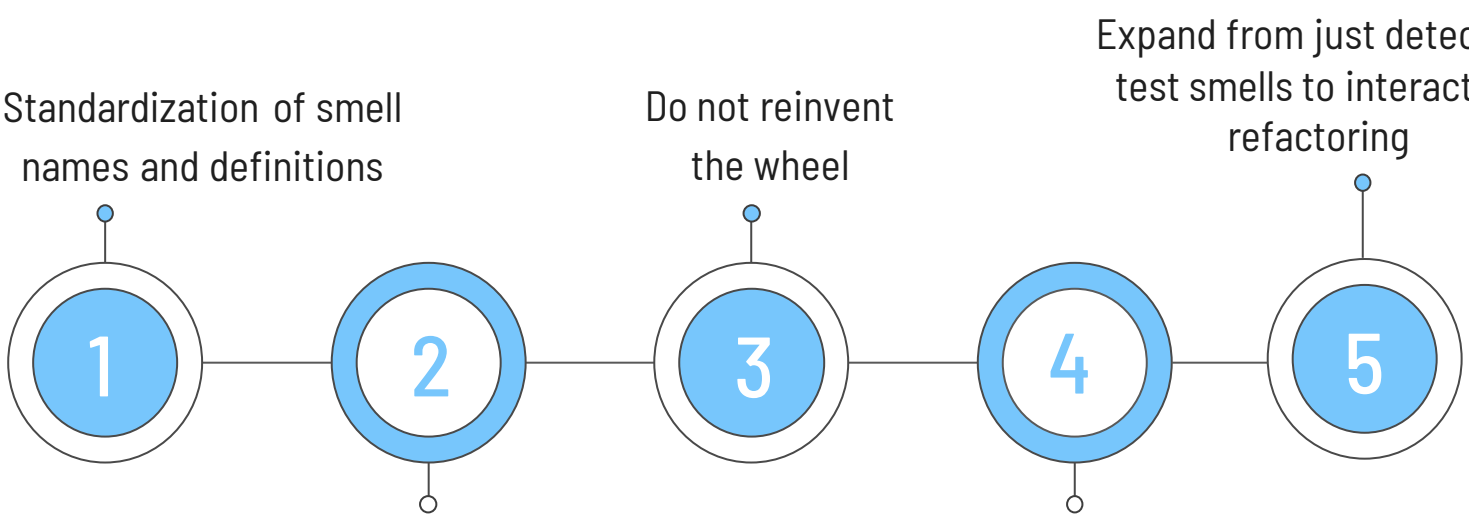
- Programming Language
- Interface
- Supported Test Framework
- Correctness
- Usages Guide Availability
- Detection Technique
- Adoption in Research Studies
- Tool Website

Search Findings

Tool	Programming Language Implemented	Programming Language Analyzed	Supported Test Framework	Correctness	Detection Technique	Interface	Usage Guide	Adoption in Studies	Tool Website
DARTS ‡ [46]	Java	Java	JUnit	F-Measure: 62%-76%	Information Retrieval	IntelliJ plugin	Yes	–	[3]
DrTest [31]	Smalltalk	Pharo ▽	SUnit	UNK	Rule Dynamic Tainting	Pharo plugin	Yes	–	[4]
DTDetector ★◇ [75]	Java	Java	JUnit	UNK	Dynamic Tainting	Command-line	Yes	–	[5]
ElectricTest [24]	Java	Java	JUnit	UNK	Dynamic Tainting	Command-line	No	–	UNK
JNose Test [70]	Java	Java	JUnit	UNK	Rule	Local web application	Yes	[71, 72]	[6]
OraclePolish ★ [42]	Java	Java	JUnit	UNK	Dynamic Tainting	Command-line	Yes	–	[7]
PoLDet [41]	Java	Java	JUnit	UNK	Dynamic Tainting	UNK	No	–	UNK
PraDeT [34]	Java	Java	JUnit	UNK	Dynamic Tainting	Command-line	Yes	–	[8]
RAIDE ‡ [60]	Java	Java	JUnit	UNK	Rule	Eclipse plugin	Yes	–	[10]
RTj ‡ [47]	Java	Java	JUnit	UNK	Rule Dynamic Tainting	Command-line	Yes	–	[11]
SoCRATES [30]	Scala	Scala	ScalaTest	Precision: 98.94% Recall: 89.59%	Rule	IntelliJ plugin	Yes	[29]	[12]
Taste [52]	UNK	Java	JUnit	Precision: 57%-75% Recall: 60%-80%	Information Retrieval	UNK	No	[54]	UNK
TeCReVis ★ [44]	Java	Java	JUnit	UNK	Metrics Dynamic Tainting	Eclipse plugin †	Yes	–	[14]
TEDD [25]	Java	Java	JUnit	Precision: 80% Recall: 94%	Information Retrieval	Command-line	Yes	[26]	[13]
TeReDetect ★ [45]	Java	Java	JUnit	UNK	Metrics Dynamic Tainting	Eclipse plugin †	Yes	–	[14]
TestEvoHound [40]	Java	Java	JUnit, TestNG	UNK	Metrics	UNK	No	–	UNK
TestHound ‡★ [39]	Java	Java	JUnit, TestNG	UNK	Metrics	Desktop application	No	–	[15]
TestLint ★ [59]	Smalltalk	Smalltalk	Sunit	UNK	Rule Dynamic Tainting	UNK	Yes	–	[16]
TestQ ★ [27]	Python	C++, Java	CppUnit, JUnit, Qtest	UNK	Metrics	Desktop application	Yes	–	[17]
TRex ‡§★ [20]	Java	Java	TTCN-3	UNK	Rule	Eclipse plugin	Yes	[49, 50, 73, 74]	[18]
tsDetect [56]	Java	Java	JUnit	Precision: 85%-100% Recall: 90%-100%	Rule	Command-line	Yes	[43, 55, 61, 64] [33, 53, 57, 62]	[19]
Unnamed [22]	UNK	Java	JUnit	Precision: 88% Recall: 100%	Rule	Command-line	No	[23, 51, 65, 66] [37, 38, 53, 58, 63]	UNK

Table 4: Characteristics of test smell detection tools.

Researh Takeaways



Replication Package

https://zenodo.org/record/4726288#.YMhyRKhKiUk

For More Information

https://testsmells.org/

Research Questions

RQ 1 What test smell detection tools are available to the community, and what are the common smell types they support?

RQ 2 What are the main characteristics of test smell detection tools?

Study Design: Search & Selection

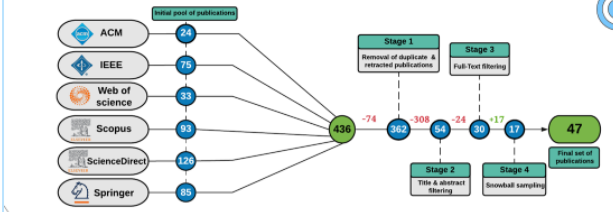


Figure 1: Overview of the volume of publications resulting from our filtering process.

Search Findings

RQ 1 What test smell detection tools are available to the community, and what are the common smell types they support?

Part 1:
Publication Years & Venues

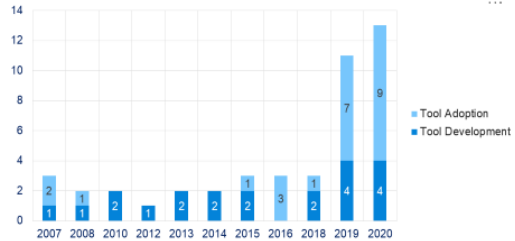
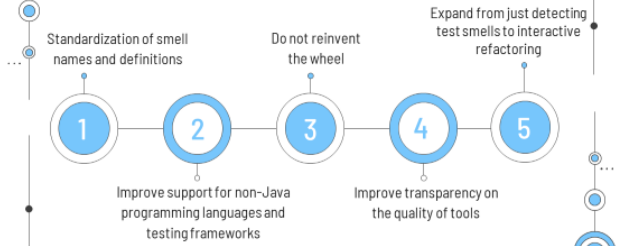


Figure 2: Yearly breakdown of tool publications.

Study Takeaways



Thanks For Watching!