

Investigating Severity Thresholds for Test Smells

Davide Spadini, Martin Schvarcbacher, Ana-Maria Oprescu,
Magiel Bruntink, Alberto Bacchelli

MSR '20: Proceedings of the 17th International
Conference on Mining Software Repositories
Pages 311–321

背景

- test smellを検出する手法が数多く研究されている
- test smellの深刻度を測定した研究は存在していない
- 優先して除去すべきtest smellがわからない

目的

- 深刻なtest smellを判定するための閾値を導出する
- 閾値を基にtest smellの深刻度を求める

test smell

- テストスイートの保守性に悪影響を与えるテストコードの書き方, 習慣
- code smellより対処が後回しになることが多い[29]

本研究で対象となるtest smell

- メソッドレベルでメトリクスを抽出できるものに限定

Table 1: Subject test smells

Test smell	Description	Problem
'Mystery Guest'	A test that uses external resources (e.g., file containing test data) [31]	Lack of information makes it hard to understand. Moreover, using external resources introduces hidden dependencies: if someone deletes such a resource, tests start failing.
'Resource Optimism'	A test that makes optimistic assumptions about the state/existence of external resources [31]	It can cause non-deterministic behavior in test outcomes. The situation where tests run fine at one time and fail miserably the other time.
'Eager Test'	A test that tries to verify too many functionalities, which can lead to difficulty in understanding the test code [31]	It is hard to read and understand, and therefore more difficult to use as documentation. Moreover, it makes tests more dependent on each other and harder to maintain.
'Assertion Roulette'	A test that has multiple assertion statements that do not provide any description of why they failed [31]	When the test fails, investigating the reason can be problematic as the assertion statement might not provide the reason why the test could have failed.
'Conditional Test Logic'	A test that has control flow statements inside a test [19]	tests can have multiple branch points and greater care must be taken when analyzing whether the test is correct.
'General Fixture'	It occurs when the test setup method creates fixtures (class fields used by the test cases) and a portion of the tests use only a subset of the fixtures [31]	The presence of this smell can result in longer execution of tests, as tests that do not use all of the fixtures still need to wait for the creation of all test fixtures.
'Empty Test'	It occurs when a test method does not contain executable statements	An empty test can be considered problematic and more dangerous than not having a test case at all since JUnit will indicate that the test passes even if there are no executable statements present in the method body.
'Magic Number Test'	A test that contains so-called "magic numbers", which are numbers used in assertions without any explanation where they come from and their value may not be immediately clear from just looking at the test code [17, 19]	The magic numbers should be replaced by a named constant, where the name describes where the value comes from or what it represents.
'Sleepy Test'	A test which contains thread suspension calls [19]	The use of this method call introduces additional delay to the test execution.
'Verbose Test'	A test that is too long and hard to understand [19]	Too long tests prevent them from being used as documentation and they are harder to maintain due to their complexity.
'Ignore Test'	A test method or class that contains the @Ignore annotation	ignored test methods result in overhead since they add unnecessary overhead with regards to compilation time, and increases code complexity and comprehension.

本研究で使用するtest smell検出ツール

- TSDETECT[23]
- Java/Junitのみ対応
- ASTベースで検出
- recallが85%

事前実験

- 開発者にtest smellをどう扱っているかについてインタビューを実施
- 筆者らが運営するBetterCodeHub(BCH)でTSDetectCTを使えるようにし、開発者にインタビュー
 - Q1. test smellは根拠があるものか
 - Q2. test smellはリファクタリングの候補となり得るか、直すだけ時間の無駄か
 - Q3. test smellの重要性は5段階のうちどれか

事前実験の結果

- test smellは誤検知または修正しない
- 実際にあがった意見
 - Conditional Test Logicでは1つくらいの分岐ぐらいでsmell扱いしないといった意見があった
 - Eager Testでも1つや2つのメソッド呼び出しは普通にあることだということでsmell扱いしないという意見があった

アプローチ

- 各test smellのメトリクスを定義
- ベンチマークベースの閾値導出
- 閾値で深刻度を求めたtest smellは開発者にとっても深刻であるかを実験

test smellのメトリクス

- 記載されていないtest smellのメトリクスは0/1で測ることができる

Table 2: Metrics used for threshold derivation, by test smell

Test Smell	Metric
Assertion Roulette	# assertions without description
Conditional Test Logic	# conditional statements
Eager Test	# production method calls
General Fixture	# unused fixtures
Magic Number Test	# magic numbers
Mystery Guest	# external files used
Resource Optimism	# files not checked for existence
Sleepy Test	# thread suspend calls
Verbose Test	# statements

ベンチマークベースの閾値導出[2]

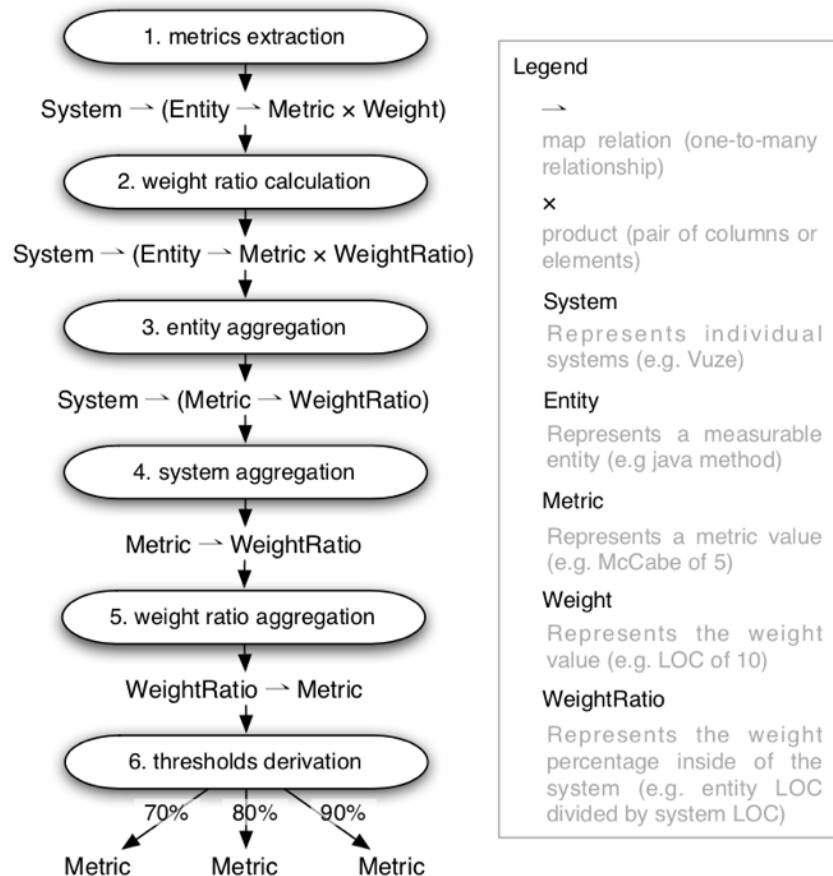


Fig. 2: Summary of the methodology steps.

- metrics extraction
 - 全てのテストメソッドでメトリクス算出
- weight ratio calculation
 - 全てのテストメソッドのWeightRatioを計算
- entity aggregation
 - メトリクス値ごとに全てのWeightRatioを集計
- system aggregation
 - WeightRatioを全てのシステムで正規化してからメトリクス値ごとに集計
- weight ratio aggregation
 - メトリクス値を昇順で並べ替え、重みの1%, 2%, ..., 100%のそれぞれの最大のメトリクス値を取得
- threshold derivation
 - 70%, 80%, 90%での閾値となるメトリクスの値を取得

Weight Ratio:

測定対象テストメソッドのLOC/
全てのテストメソッドのLOC合計

求めた閾値の例

- Eager Testの例

- メトリクス：関数呼び出し回数
- 閾値90%は39, 90%のテストメソッドは少なくとも39回のメソッド呼び出しがある

開発者による評価実験

- 求めた閾値を組み込んだTSDetectをBCHで使えるようにする
- 深刻度を提示されたtest smellを開発者が見てどう感じたかをインタビュー

インタビュー項目

- この**test smell**がテストへの保守性に与える悪影響を以下の中から選んでください
 - 知らない，誤検知，かなり低い，低い，中程度，高い，かなり高い
- この**test smell**に対してどのような処置を行いますか．以下の中から選んでください
 - 今すぐリファクタリングする，技術的負債として後でリファクタリングする，無視する/誤検知

インタビュー項目

- リファクタリングにかかった工数を答えてください（任意）
- 他に意見があれば教えてください（任意）

閾値の導出結果

- 90%未満の段階で0のtest smellは1でVery Highの深刻度がつけられる

Table 3: Severity thresholds calculated from the benchmark

Test Smell	Severity Threshold		
	Medium	High	Very High
Assertion Roulette	3	5	10
Eager Test	4	7	39
Verbose Test	13	19	30
Conditional Test Logic	0	1	2
Magic Number Test	0	0	1
General Fixture	0	0	0
Mystery Guest	0	0	0
Resource Optimism	0	0	0
Sleepy Test	0	0	0

閾値の妥当性の評価

- スピアマンの順位相関検定で閾値で求められた深刻度と開発者の評価の相関の有無を調査

Table 5: Spearman's rank correlation between test smell severities as set by our thresholds and rated by users. Statistically significant results are in bold.

Test Smell	Responses	p	r_s
Eager Test	42	0.027	0.342
Conditional Test Logic	36	<0.001	0.753
Verbose Test	51	<0.001	0.679
Assertion Roulette	47	0.016	0.350

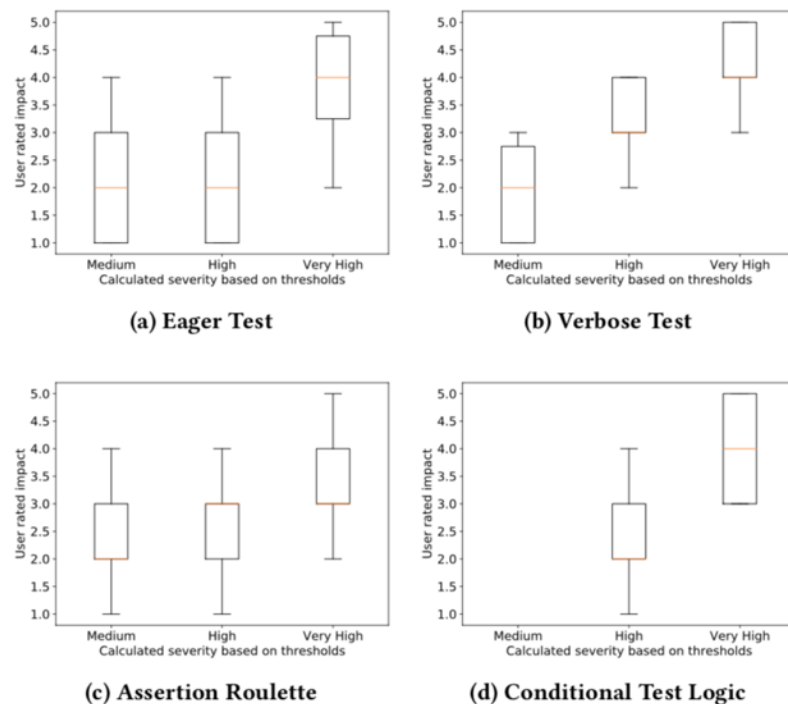


Figure 4: Test smell severity vs. user rated impact values

開発者がとる処置の集計結果

- 多くのtest smellでリファクタリングすべきという回答が多く見られた

Table 6: Actions taken by users, by identified test smell

Test Smell	Responses	Immediate Refactor	Long-term Refactor	No Action
Empty Test	14	71.43%	21.43%	7.14%
Sleepy Test	13	61.54%	23.08%	15.38%
Mystery Guest	10	60.00%	10.00%	30.00%
Resource Optimism	12	58.33%	16.67%	25.00%
Cond. Test Logic	39	53.85%	33.33%	12.82%
Verbose Test	54	48.15%	40.74%	11.11%
Magic Number Test	23	47.83%	34.78%	17.39%
Assertion Roulette	55	29.09%	50.91%	20.00%
Eager Test	51	21.57%	50.98%	27.45%
Ignored Test	10	20.00%	50.00%	30.00%
General Fixture	20	20.00%	40.00%	40.00%

まとめ

- **test smell**のメトリクスの閾値を求め、深刻度を示した
- 閾値を元に得られた**test smell**の深刻度は開発者が感じる深刻度と相関が認められた
- 深刻度を示すことで開発者が**test smell**に処置を与えるべきであると考えることが増えた

所感

- **test smell**は割と昔から定義されているが、本研究のような深刻度を求める研究がなかったのは驚き
- **BHC**のようなサービスを運営していなければ難しいような研究だと思った
- 処置の集計で深刻さを提示しない場合と比較して欲しかった