

Automated Static Analysis of Unit Test Code

Rudolf Ramler, Michael Moser and Josef Pichler

2016 IEEE 23rd International Conference on Software
Analysis, Evolution, and Reengineering

発表者 吉富 楓雅

目次

- 研究の目的とリサーチクエスチョン
- 研究結果
- 結論と今後の展望

研究の目的と リサーチクエスチョン



テストコードの進化と信頼性

- 多くのプロジェクトでは, 大量のテストコードが開発中にプロダクトコードとともに進化している[2]
 - * テストコードとプロダクトコードの比が1 : 1と報告されている[3]
- 他のコードと同様に, テストコードもバグを含む[4]
 - * このような不具合は, 自動テストの有効性, 信頼性, 有用性に悪影響を及ぼす

• [2] A. Zaidman, B. Van Rompaey, A. van Deursen, and S. Demeyer, “Studying the co-evolution of production and test code in open source and industrial developer test processes through repository mining,” *Empirical Software Engineering*, 16(3), pp. 325-364, June 2011.

• [3] A. van Deursen, L. Moonen, A. van den Bergh, and G. Kok, “Refactoring test code,” 2nd Int. Conf. on Extreme Programming and Flexible Processes in Software Engineering (XP 2001), 2001.

• [4] A. Vahabzadeh, A. Milani Fard, and A. Mesbah, “An Empirical Study of Bugs in Test Code,” *IEEE Int. Conf. on Software Maintenance and Evolution (ICSME 2015)*, 2015.

研究の目的

- 潜在的なコーディングとメンテナンスの問題を特定するために、自動静的分析が使用されている[5]
- 本論文では、単体テストコードの品質を改善するための自動静的解析について報告している
- JUnitフレームワークをベースにしたテストに焦点を当てる

- [5] F. Wedyan, D. Alrmuny, and J. M. Bieman, “The effectiveness of automated static analysis tools for fault detection and refactoring prediction,” Int. Conf. on Software Testing Verification and Validation (ICST'09), 2009.

JUnitとは

- Java言語によって開発されたプログラムに対して単体テストを行うためのアプリケーションフレームワーク
 - * Kent Beckらによって開発された
- 単体テストが簡単に行えるようになり, 開発に要する時間を短縮して効率化を測ることができる

The logo for JUnit, with the 'J' in green and 'Unit' in red.

リサーチクエスチョン

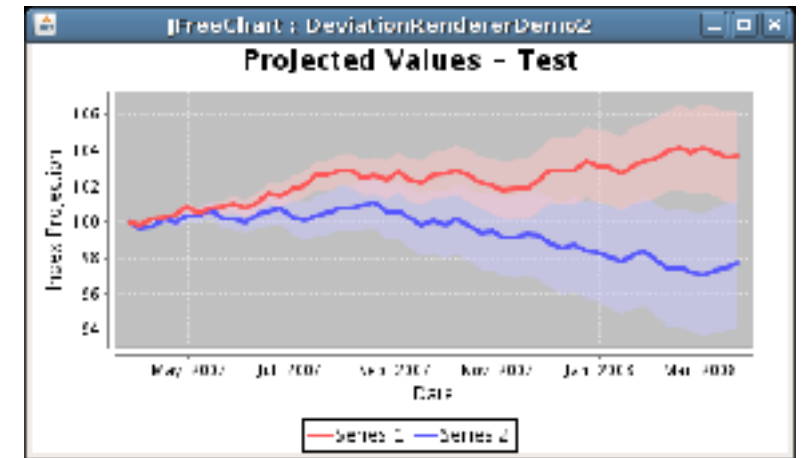
テストに関連するコーディングの問題を明らかにするために、自動解析を使用するのは効果があるのか？

研究結果

RESULTS



分析対象



- JFreeChart5(ver 1.0.12)

* Java用のオープンソースチャートライブラリ

- ▶ 2000年開始以来30万行もののJavaコードを開発した
- ▶ 約500個のJUnitのテストクラスで7000を超えるテスト方法によってほぼ正確にテストが行われる
- 今回はorg.jfree.chart.junitパッケージのテストを分析する

JFreeChartでの問題

- 旧バージョンを使っている部分が多い
 - * プロジェクトの新しい部分はずでにJUnit4.xを使用しているが、多くは依然としてJUnit3.81を使用
- 単体テストにおける問題に悩まされている
 - * 例
 - * #1119: 異なる順序で実行された場合に、いくつかの単体テストが予期せず失敗する

指標の概要

TABLE I. METRICS AND RULE VIOLATIONS FOR *ORG.JFREE.CHART.JUNIT*

File	LOC	NOM	NOA	Violations	A	N	L	S	M
1	328	3	63	140	129	9	2		
2	409	4	46	112	95	13	4		
3	240	7	21	67	45	16	6		
4	222	6	20	60	43	11	6		
5	195	4	16	53	33	15	5		
6	162	3	8	33	18	9	6		
7	154	3	8	30	14	12	4		
8	210	4	6	27	11	12	1	1	2
9	222	4	6	27	11	12	1	1	2
10	233	4	6	27	11	12	1	1	2
11	228	4	6	27	11	12	1	1	2
12	228	4	6	27	11	12	1	1	2
13	227	4	6	27	11	12	1	1	2
14	227	4	6	27	11	12	1	1	2
15	233	4	5	27	11	12	1	1	2
16	213	3	5	26	8	11	4	1	2
17	71	3	5	26	8	11	4	1	2
18	178	3	5	26	8	11	4	1	2
19	210	3	5	26	8	11	4	1	2
20	214	3	5	26	8	11	4	1	2
21	210	3	5	26	8	11	4	1	2
22	210	3	5	26	8	11	4	1	2
23	327	5	5	28	13	10	2	1	2
24	173	2	3	19	7	8	1	1	2
25	146	1	2	15	5	7		1	2
26	208	3	3	14	7	4	1	1	1
27	86	1	2	7	5	2			
28	103	1	1	6	1	3	2		
Total	5867	96	280	982	559	292	75	19	37
					57%	30%	8%	2%	4%

LOC: Lines of Code

コードの行数

NOM: the Number of test Methods

テストメソッドの数

NOA: the Number of Assert statements

アサートステートメントの数

A:アサーション

N:命名規則違反

L:テストコード

S:セットアップ

M:その他

参考: アサーションとは

- プログラミング言語などの仕様・機能の1つ
 - * プログラムの前提として満たされるべき条件を記述
 - * 実行時にそれが満たされない場合はエラーや例外を発生させる

表からわかること

- 違反の数はアサートステートメント（NOA）の数と密接に関連する
 - * 検出された違反のほとんどがアサーション（A：57%）に関連しているため
 - * 命名規則違反（N：30%）
 - * テストコード（L：8%）
 - * セットアップ（S：2%）
 - * その他（M：4%）

違反の分類

- Confirmed Faults: 確認されたフォルト
 - * 多くの場合, JUnitのアサーションメソッドが誤って使用され, アサーションに関連する違反が多く発生する
- Low priority violations: 優先度の低い違反
 - * 違反ではなく「推奨」として優先順位を付ける必要があるものが存在する

違反の分類

- Irrelevant results: 関連のない結果
 - * プロジェクトが標準の命名規則とは異なるものを使用するために多くが無関係とみなされることがある
- Unintentional discoveries: 意図しない発見
 - * 検出すべき問題とは異なるものに関連してしまうものがある

結論と今後の展望



結論

- JFreeChartは十分にテストされたシステムにも関わらず違反が数多く発見された
 - * 1つのテストパッケージに対して982件の違反は驚くべき数
- 自動解析を使用することによって、これらを効率的かつ効果的に発見することができた
 - * これらの発見はレビューによって行われるべきであるが、手作業のために時間がかかってしまう

今後の展望

- 今回は最初の分析としてJFreeChartを使用した
 - * 今回の結果をもとにルールセットを微調整し, 最終的にはオープンソースプロジェクトを包括的に評価できるようにしたい
- 業界で使用されているプロジェクトのテストコードについて
 - * 今回で使われた自動分析のルールの適用

所感



所感

- 今回の研究がオープンソースに一般化されたら, 違反をより分析しやすくなるので, テスト研究が進んでいきそう
- 単体テストの問題点についての解決策などは言及していなかったなので, 言及している論文を探して読みたい
- テストの違反分類は, 自身の研究に役立ちそう