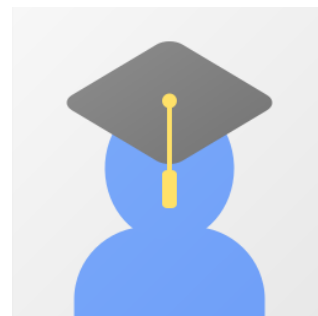
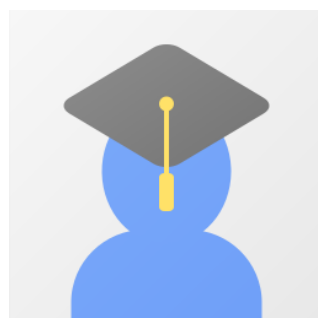


Empirically Revisiting the Test Independence Assumption



Sai Zhang, Darioush Jalali, Jochen Wuttke, Kivanç Muslu, Wing Lam,
Michael D. Ernst, David Notkin

In ISSTA 2014, Proceedings of the 2014 International Symposium on Software Testing and Analysis,
(San Jose, CA, USA), July 2014, pp. 385-396.

発表者 吉富 楓雅

目次

- 依存テスト
- 依存テストの調査結果
- 依存テスト検出ツールDTDetector
- 評価実験と結果
- まとめと所感

要約

テストスイートでは、すべてのテストケースが独立している必要がある。しかし実際にはテスト依存性によってテストの独立性は失われている。また、テストの依存関係はあまり研究されていない。本論文では、

(I) 依存テストがどのような影響を及ぼすのかを調査し、

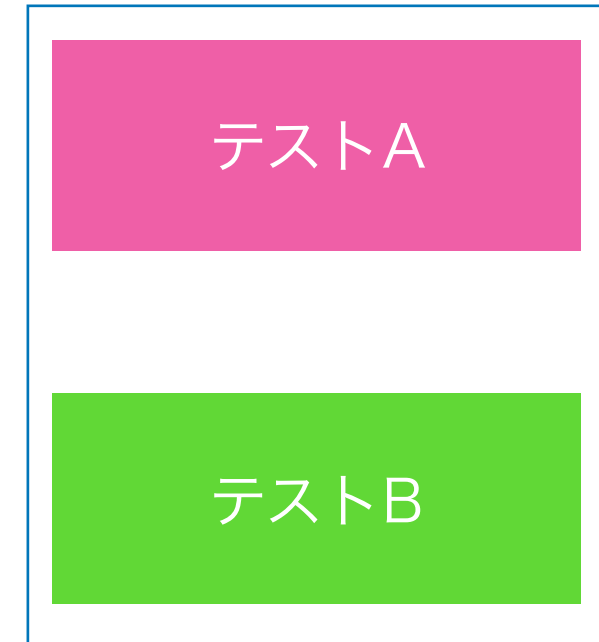
(II) 依存テストを検出するアルゴリズムとそれを実装したツールDTDetectorを提案し、評価実験を行う。

依存テスト

順序依存テスト

- 2つのテストAとBを含むテストスイート
 - * テストAを実行してからBを実行：テストはPass
 - * テストBを実行してからAを実行：テストはFail
- テストBを実行した順によってテストAの結果が異なる場合がある
- このようなテストAを、**順序依存テスト**と呼ぶ

テストスイート



Pass



Fail

テストの独立性

- **テストは他のテストに影響を及ぼしてはならない**
 - * テストを任意の順序で実行しても同じ結果が得られるようにしなければならない
- 多くのテスト優先順位付けに関する論文で明示的にテストは独立していると仮定している
 - * 2000年から2013年, ICSE, FSE, ASEなどにテスト優先順位付けに関する31件の論文が掲載された[1].
 - ▶ 27件の論文が明示的にテストの独立性を前提としていた

[1]S. Zhang and K. Muşlu. List of papers on test prioritization techniques from major software engineering conferences and journals, 2000–2013.
http://homes.cs.washington.edu/~szhang/test_prioritization_paper_list_2000-2013.txt.

依存テストの発生原因

- 他のテストとの相互作用に起因
 - * 開発者がコードを記述する際に、**実行環境や使用するリソースを初期化し忘れる**などのミスが原因
 - ▶ 実行環境：グローバル変数の値、ファイルの内容など
- 依存テストが存在すると、別のテストによって実行環境が変更された時に元の実行環境を無効にする可能性がある

貢献

- 開発者にとって有用
 - * 依存テストは**偽陽性や偽陰性の発生原因になる可能性**
があるため、無視できない問題であると分かる
- 開発者はどのコーディングパターンがテスト依存を引き起こすのかを知ることができる

問題と目的

- 問題点

- * 既存研究では、依存テストによって及ぼされる影響について調査していない

- 研究の目的

- * 依存テストが及ぼす影響を分析する
 - * 依存テストを抽出するためのツールを提案し様々なプロジェクトで評価実験を行う

依存テストの調査結果

各プロジェクトの依存テスト

問題追跡システム

依存テスト

依存を明らかにするために必要なテストの数

Issue Tracking System	Dependent Tests				# Involved Tests for Manifestation				
	Total number	Severity 重大度			Self	1 test	2 tests	3 tests	Unknown
		Major	Minor	Trivial					
Apache	26	22	3	1	0	5	18	1	2
Eclipse	59	0	59	0	0	0	49	1	9
JBoss	6	6	0	0	0	0	3	0	3
Hibernate	3	1	1	1	0	0	3	0	0
Codehaus	2	2	0	0	1	1	0	0	0
Total	96	31	63	2	1	6	73	2	14



依存テストの合計数

表1：各プロジェクトの依存テスト

各プロジェクトの依存テスト

開発者が依存テストを解決した方法

問題追跡システム

根本原因

Issue Tracking System	Resolution					Root Cause			
	Days	Patch Location				Static variable	File system	Data-base	Un-known
		Code	Test	Doc	Unfixed				
Apache	93	5	20	0	1	9	3	8	6
Eclipse	48	1	8	49	1	49	0	0	10
JBoss	44	0	2	0	4	1	0	0	5
Hibernate	6	0	1	0	2	0	0	2	1
Codehaus	3	0	1	0	1	0	1	0	1
Total	194	6	32	49	9	59	4	10	23

開発者が依存テストを解決するために要した平均日数

表1：各プロジェクトの依存テスト

依存テストがもたらす影響

- 依存テストは**False alarm**あるいは**Missed alarm**を引き起こすことがある
 - **False alarm**
 - * テストがパスする必要があるが、依存テストによって誤ったエラーが発生する
 - **Missed alarm**
 - * 依存テストがプログラムのバグを隠してしまうことがある

依存テスト検出ツール

DT Detector

依存テストの検出

- 前提：依存テスト検出問題の一般的な形は**NP完全**であるため、完全に解決する効率的なアルゴリズムを見つけることは期待していない
 - * NP (Non-deterministic Polynomial time；非決定性多項式時間)
- 正確な解を近似するため、すべての依存テストのサブセットを見つける4つのアルゴリズムを示す

依存テスト検出アルゴリズム

- Reversal Algorithm
- Randomized Algorithm
- Exhaustive Bounded Algorithm
- Dependence-Aware Bounded Algorithm

ツールの実装

- **DTDetector**というツールで、4つの依存テスト検出アルゴリズムを実装
 - * JUnit 3.x / 4.xをサポート
- 実行後に作成された一時ファイルを削除するなど、**リソースをリセットさせる**
- ファイルレベルの読み取り/書き込み情報を提供するJava SecurityManagerをインストールすることによって、**各ファイルのアクセスを監視**
- **記録された読み取り/書き込み情報を使用してテストの依存性を検出**

評価実験と結果

リサーチクエスチョン

1. 各検出アルゴリズムがプログラムで検出できる依存テストはどの程度か？
2. 各アルゴリズムが依存テストを検出するのにどの程度の時間が必要か？
3. 依存テストはテストの優先順位付けに影響するのか？

評価対象

Program	LOC	Tests	Auto Tests	Revision
Joda-Time	27183	3875	—	b609d7d66d
XML Security	18302	108	665	version 1.0.4
Crystal	4676	75	3198	1a11279d3d6c
Synoptic	28872	118	2467	d5ea6fb3157e

自動テスト生成ツールRandoop[2]によって生成されたテストの数

表 2 : 評価対象のプログラム

RQ1 · RQ2

表2から取得したテストの総数

各テストスイートで検出された依存テストの総数

分析にかかった時間(秒)

Subject Programs	# Tests	# Detected Dependent Tests								Analysis Cost (seconds)							
		Rev	Randomized			Exhaustive		Dep-Aware		Rev	Randomized			Exhaustive		Dep-Aware	
			10	100	1000	$k=1$	$k=2$	$k=1$	$k=2$		10	100	1000	$k=1$	$k=2$	$k=1$	$k=2$
Human-written unit tests																	
Joda-Time	3875	2	1	1	6	2	≥ 2 *	2	≥ 2 *	11	57	528	5538	1265	4×10^6 *	291	5×10^5 *
XML Security	108	0	1	4	4	4	4	4	4	11	65	594	5977	106	11927	93	3322
Crystal	75	18	18	18	18	17	18	17	18	2	14	131	1304	166	7323	95	4155
Synoptic	118	1	1	1	1	0	1	0	1	1	7	67	760	25	3372	24	1797
Total	4176	21	21	24	29	23	≥ 24	23	≥ 25	26	143	1320	13579	1562	4×10^6 *	503	5×10^5 *
Automatically-generated unit tests																	
Joda-Time	2639	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
XML Security	665	138	167	171	171	129	≥ 129 *	128	≥ 128 *	6	50	430	4174	133	1×10^5 *	128	5×10^4 *
Crystal	3198	75	159	162	164	55	≥ 55 *	55	≥ 55 *	18	103	949	9436	2477	8×10^6 *	2297	1×10^6 *
Synoptic	2467	3	3	7	10	2	≥ 2 *	2	≥ 2 *	12	81	770	6311	454	1×10^6 *	454	2×10^4 *
Total	8969	216	329	340	345	186	≥ 186	185	≥ 185	36	234	2149	19921	3064	1×10^7 *	2879	1×10^6 *

表3：検出された依存テスト数と分析に要した時間

テスト優先順位付け

Label	Technique Description
T1	Randomized ordering
T3	Prioritize on coverage of statements
T4	Prioritize on coverage of statements not yet covered
T5	Prioritize on coverage of methods
T7	Prioritize on coverage of functions not yet covered

※各テスト優先順位付け方法の詳細については[3]を参照

表4：実験で使用するテスト優先順位付け方法

RQ3

Subject Program	T1	T3	T4	T5	T7
Joda-Time	0	0	1	0	0
XML Security	0	0	0	0	0
Crystal	12	11	16	11	12
Synoptic	0	0	0	0	0
Total	12	11	17	11	12

- ・依存テストに優先順位付けを行った時と，行わなかった時に異なるテスト結果となった数
- ・偽陽性や偽陰性が発生するのを承知で最適化を行っても，改善することができなかった

表5：テストの優先順位付けへの影響

まとめと所感

まとめ

- 2014年には依存テストの影響は研究されていなかった
 - * 依存テストの悪影響を調査したのはこの論文が初
 - ▶ 依存テストが存在しても問題がないと言われていた
- 提案ツールによって依存テストを抽出できた
 - * 各アルゴリズムによって検出数やコストは異なる
- 優先順位付けを行った際に依存テストは影響を及ぼす
 - * 依存テストによる影響を最適化することはできなかった

所感

- Missed Alarmって偽陰性のこと？
 - * Missed alarm
 - ▶ 依存テストがプログラムのバグを隠してしまうことがある
- テスト優先順位付けが勉強不足なので，[3]の論文を読みたい