

SIGSS発表原稿 作成用スライド

SIGSS 7 月研究会@帯広

和歌山大学システム工学部システム工学科

吉富 楓雅

論文タイトル（仮）

プロダクトコードとテストコード の 積集合の調査

要約

本論文の目次

- 目次

- * 第2章：背景

- * 第3章：分析手法

- * 第4章：分析結果

- * 第5章：結論

背景

ソフトウェアテストとは

- ソフトウェアテストとは

*完成したソフトウェアが正しく動作するか,完成したソフトウェアに欠陥が混入していないかを確認するための作業である.オープン ソースソフトウェア (OSS) 開発における品質保証活動は,テスト活動よりも短期間でリリースを行い,リリース時にOSSプロジェクトによって公開されたソース コードを不特定多数の開発者や利用者が検証, 欠陥報告し, 開発者が欠陥を修正する形態が主流である.OSS 開発でテスト活動が十分に実施されていない理由として, テスト活動に関心のある開発者の不足やソフトウェアテストを熟知してる開発者の不足が挙げられる.このような限られたリソースにおけるテスト活動は困難である [1][2].

- [1] Beller, M., Gousios, G., Panichella, A. and Zaidman, A.: When, How, and Why Developers (Do Not) Test in Their IDEs, *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pp. 179–190 (2015).
- [2]Tomotaka Minami :Master’s Thesis, Department of Information Science, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT1551105, March 16, 2017.

テストコードの発達

- 様々なオープンソースプロジェクトの開発が進むなか、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[1]
- プロダクトコードとテストコードの比が1:1とも言われている[2]
- つまり、プロダクトコードが開発されるにつれ、テストコードにもバグが混入する可能性が高まる

- [1] 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.
- [2] 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.

テストをテストする

- プロダクトコードに対して、テストコードは一般的に1～2倍の量を記述する
- テストが適切であるかを調べるためには、この2倍の量のテストを行わなければならないのか？
 - * 現場において、そのようなことは不可能
- では、テストのテストはどのようにすれば効率よく適切に行うことができるのか？

最悪のケース

- ソフトウェアテストにおいて 一番最悪なケースは, テストコードの誤りによってプロダクトコード内のバグが見逃されてしまうことである
 - * 開発者が頑丈なテストケースを作成したと思っても, 何かしらのミスでテストコードに綻びが生じていることは予想できる
 - * 柏先輩談: 実際に企業においても, プロダクトコードに対して書かれた膨大な量のテストコードを誰がテストするのかについて疑問を投げかける者がおり, またプログラマ自身も, 誤っている可能性があるコードに対して, テストコードが完全に網羅されているのか不安になりながらコード作成しなければならず, 心理的な負担も大きい.

研究の目的

- テストを改善することや、強固にするための研究は行われているが、テストに誤りが起こることに対しての研究は行われていない
 - * テストは人間が作るので、バグを全く含まずに作成することは不可能。
 - ▶ いくら強固に作っても、必ずどこかに綻びが生じていることは予想できる
- 開発者が安心してバグを見つけるためのテストに誤りがあれば、心理的な負担も大きい。
 - * どのような場合に綻びが生じやすいのか予め分かっているならば、テスト作成の際にそれを加味しながら作成することができる
- 本研究では、そのようなケースが(あるとは思うが)存在するのかを検証し、どのような事例の時に発生しやすいのかについて分析を行う。

研究による展望

- 先ずはテストコードにそのような欠陥があるのかどうかを調査し、発見できた場合、それがどのような分類できるのかを検証することを目標とする。
- この研究が完成すれば、企業などでソフトウェアテストをする際に、テストにおける最悪の失敗（テストの謝りによってバグが見逃される）を事前に知っておくことが可能となり、プログラマ自身が安心してコードを打つことができるであろう。

分析手法

データのサンプリング

- 分析対象



- * Mybatis-3

- ▶ **MyBatis** :カスタム SQL, ストアドプロシージャ, 高度なマッピング処理に対応した永続化フレームワーク

- 150のテストデータをサンプリング
 - * 条件: バグIDに紐づけられており, プロダクトコード・テストコード共に修正あるいはコードの追加が見られるもの
- このうち, テストバグによってプロダクトコードのバグが見逃されているものを発見する

分析方法

- Mybatis-3のバグ IDに対応しているコミットをsourcetreeのrepo-toolsより検出。全てのコミットの中からバグIDに対応しているもののみを抽出し、これをgithubで分析。このうちテストコードに何らかの変更・修正が見られるものを抽出対象とし、それ以外のものは除外する。
- * プロダクトコードの変更が見られないコミットも存在したが、テストを作るということはプロダクトコードに何らかの不具合があると予想して開発者が作成するのであり、プロダクトコードを変更したコミットは別に存在すると仮定し、分析対象とする
 - ▶ ※repo-toolsの説明は必要なのか？

分析方法

- 目視で確認する

- * 見るべきもの

- ▶ バグID、コミット日時、コミットID、コミットメッセージ、ファイルパス、ファイル変更とテスト変更の有無、変更点の詳細、テストコード・プロダクトコードのどちらから先に変更されたのか？、テストコードの誤りによってプロダクトコードの誤りが見逃されている可能性があるか？

- Excel（表形式）にまとめた

分析結果

Excel ファイル 編集 表示 挿入 フォーマット ツール データ ウィンドウ ヘルプ

プロダクトコードとテストコードのバグ修正検証 — 自分の Mac に保存済み

ホーム 挿入 ページレイアウト 数式 データ 校閲 表示 開発 テーブル

文字列の折り返し セルを結合して中央揃え

標準 美字付き書式 テーブルとして書式設定 セルのスタイル 挿入 削除 書式 並べ替え/フィルター

C5 X ✓ fx バグID

	A	B	C	D	E	F	G	H
1	プロダクトコードとテストコードのバグ修正検証							
2	mybatis-3							
3	検索条件: プロダクトコード・テストコード共に修正・削除・コード追加が見られるものを抽出すること							
4								
5		番号	バグID	バグIDのURL	コミット日時	コミットURL	コミットID	コミットメッセージ
6		1	#411	https://github.com/mybatis/mybatis-3/issues/411	25-Aug-15	https://github.com/mybatis/mybatis-3/commit/60157ab556e36dbb83301d7d032ae799d6ace167	60157ab556e36dbb83301d7d032ae799d6ace167	Close #411 with a b See the previous c
7		2	#782	https://github.com/mybatis/mybatis-3/pull/782	13-Aug-17	https://github.com/mybatis/mybatis-3/commit/e5435a63fa69f77574c380d8fda452e77df6072f	e5435a63fa69f77574c380d8fda452e77df6072f	Reverted #782 workaround for #
8		3	#268	https://github.com/mybatis/mybatis-3/pull/268	14-Sep-14	https://github.com/mybatis/mybatis-3/commit/65f432b98c074fcc57304dec83e234f77690ac0e	65f432b98c074fcc57304dec83e234f77690ac0e	Merge pull request # hazendaz/ma
9		4	#841	https://github.com/mybatis/mybatis-3/pull/841	20-Nov-16	https://github.com/mybatis/mybatis-3/commit/5f87a8fa8cc0e86d1ed1eb311b089cc80	5f87a8fa8cc0e86d1ed1eb311b089cc808b2e1bb	Merge pull request # kazuki43zoo/sup
10		5	#986	https://github.com/mybatis/mybatis-3/pull/986	19-Apr-17	https://github.com/mybatis/mybatis-3/commit/582ee9405ea72bd95ad144da56da7bab	582ee9405ea72bd95ad144da56da7bab394eace7	Merge pull request # hazendaz/ma
11		6	#976	https://github.com/mybatis/mybatis-3/pull/976	17-Apr-17	https://github.com/mybatis/mybatis-3/commit/4d57247bcd6873c0d4113c09166ce6b9f9b8bb30	4d57247bcd6873c0d4113c09166ce6b9f9b8bb30	Merge pull request # harawata/typehan
12		7	#411	https://github.com/mybatis/mybatis-3/issues/411	25-Aug-15	https://github.com/mybatis/mybatis-3/commit/263b47e59ff7e06c8a5d79b435d98ad1d0795a07	263b47e59ff7e06c8a5d79b435d98ad1d0795a07	BoundSql#hasAdditio eter should check
13		8	#1057	https://github.com/mybatis/mybatis-3/issues/1057	27-Jul-17	https://github.com/mybatis/mybatis-3/commit/50a38a1eac9f9977bc999fb78d6d6c163903f44	50a38a1eac9f9977bc999fb78d6d6c163903f44	Allow replacing conf properties o
14		9	#842	https://github.com/mybatis/mybatis-3/pull/842	24-Nov-16	https://github.com/mybatis/mybatis-3/commit/a8a8f1f8c874aa306a4eb4c395c7cd800ec85587	a8a8f1f8c874aa306a4eb4c395c7cd800ec85587	Merge pull request # kazuki43zoo/add-a
15		10	#1030	https://github.com/mybatis/mybatis-3/issues/1030	11-Jun-17	https://github.com/mybatis/mybatis-3/commit/db601b8ef70fe8d5be91edc855637e798	db601b8ef70fe8d5be91edc855637e798a3f0da3	Allow long type c parameter when us
16		11	#39	https://github.com/mybatis/mybatis-3/issues/39	23-Dec-15	https://github.com/mybatis/mybatis-3/commit/72d469532f634c7badccdbbcc7f73696b73c5927	72d469532f634c7badccdbbcc7f73696b73c5927	fix for #39. No need objects with rand
17		12	#947	https://github.com/mybatis/mybatis-3/pull/947	16-Mar-17	https://github.com/mybatis/mybatis-3/commit/54fe9da4aabfcdcfce93f219797082894b506d164	54fe9da4aabfcdcfce93f219797082894b506d164	Support enum

Mybatis-3 +

コマンド

125%

分析結果

- テストコードの誤りによってバグコードが見逃されているものは確認できなかった（5月13日地点）
 - * Mybatis-3は現在はアクティブなプログラムであり、テストコードにバグが混入することは考えにくい
- 今回のバグレポートは、プログラムの仕様に対してのコード変更が多かった
 - * 例：エラープロンプトの明確性・バージョンアップによる修正・Mybatisのプログラムで使用できない項目の削除など

分析方法の補正

- バグ修正に焦点を充てるべきだったのでは？
 - * バージョンアップに対する変更からは、テストにバグが混入するとは思えなかった
 - ▶ コミットのタイトルにBugfixなどが明記されているものを選択すべき？
- テストを追加しているだけのものは、テストにバグが混入していることを見抜けるか疑問
 - * しかし、テスト追加時にバグも混入する可能性があるので、見るべきではあると思う
 - * 逆に、テストを修正した際にもテストコードに誤りが混入する可能性がある

結論

今後の分析対象について

- ある程度開発が落ち着いたものを分析対象とする。テストコードにバグが見られるものをデベロッパが発見することはある程度期間が過ぎてからでないと予想される
- * 個人的見解：開発中は目の前の問題を開発するのに必死なはず。開発中にテストをテストすることはしていないと思う
- コミットの数が多いものを探す。開発者とプロダクトコードが多い方が分析対象が見つけやすい。

(関連研究)