



ソフトウェアテストの偽陰性に関するケーススタディ

2018年10月28日 (金)

オープンソースソフトウェア工学研究室

発表者 吉富 楓雅

目次

- 既存研究
- 分析方法とリサーチクエスション
- 欠陥報告の例
- 現状と今後の予定

要約

テストコードに欠陥が混入し
プロダクトコードに欠陥があ
るにも関わらずテストが成功
する現象をソフトウェアの偽
陰性と呼ぶ。既存研究では偽陰
性 (Silent Horror) を引き起
こした欠陥の種類については
明らかにしているが、発生原
因については明らかにしてい
ない。本研究では、Apache
Derbyプロジェクトを対象に
テストコードの不具合報告を
目視し、テストコードの欠陥
発生原因の分析を行う。

ソフトウェアテスト

- ・ ソースコードが期待通りに機能するか検証するために実施
- ・ しかし、**作成したテストコードには欠陥が混入する可能性がある**
 - * 多くの開発者がテストコードはプロダクトコードよりも欠陥を含む可能性が高いと述べている[1][2]
- ・ テストコードに欠陥が混入すると,
 - * **欠陥が見逃される**
 - * **プロダクトコードに問題がないにもかかわらずテストが失敗する**

[1]C. Jones. Programming Productivity. McGraw-Hill., New York, NY, 1986.

[2]S. McConnell. Code Complete. Microsoft Press., Redmond, WA, 1993.

テストコードの欠陥による問題

- Vahabzadehらは2つに分類[1]

- * Silent Horror(偽陰性・False Negative) :

- ▶ プロダクトコードに欠陥があるが、テストが成功する
 - テスターが発見しにくいので、手戻りコストがかかる

- * False Alarm(偽陽性・False Positive):

- ▶ プロダクトコードに欠陥はないが、テストが失敗する
 - テストが失敗している原因を探すことにコストがかかるが、品質には問題がない

既存研究

[1] An Empirical Study of Bugs in Test Code

Arash Vahabzadeh, Amin Milani Fard, Ali Mesbah

The 31st International Conference on Software Maintenance and
Evolution(ICSME2015)

プロジェクトにおけるテストコードの欠陥件数

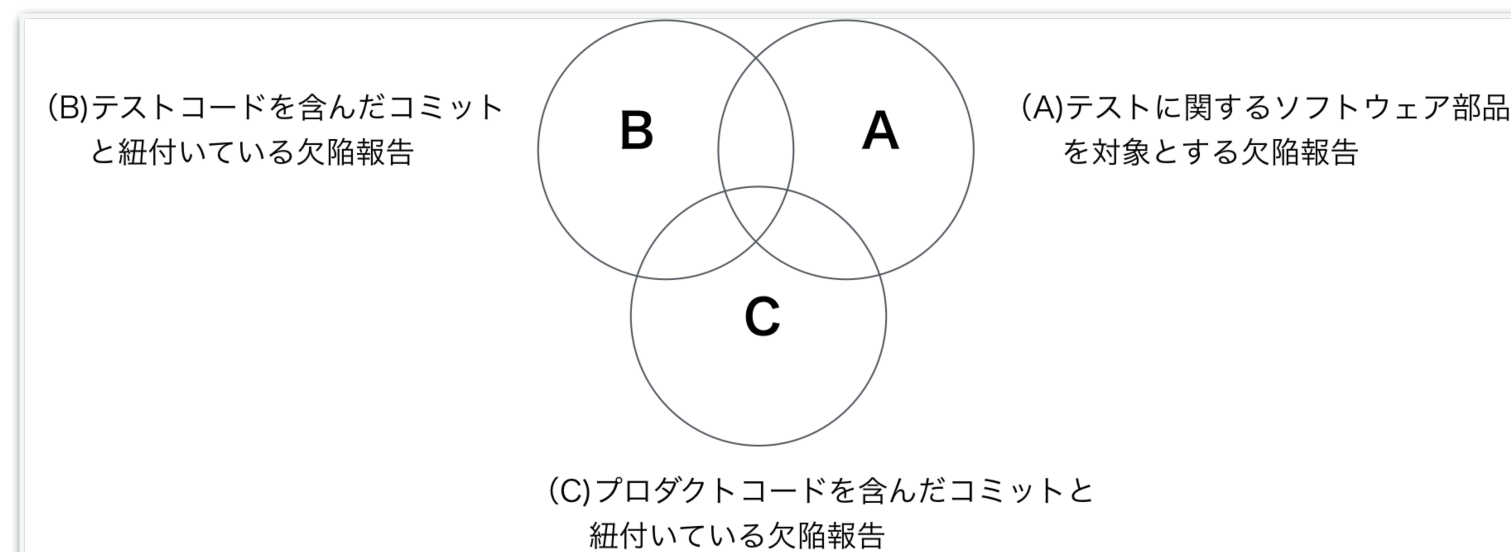
TABLE II: Top 10 ASF projects sorted by the number of reported test bugs.

Project	Production Code KLOC	Test Code KLOC	# Test Bug Reports
Derby	386	370	614
HBase	587	195	440
Hive	836	124	295
Hadoop HDFS	101	57	286
Hadoop Common	1249	380	279
Hadoop Map/Reduce	60	24	231
Accumulo	405	78	187
Qpid	553	93	152
Jackrabbit Content Repository	247	107	145
CloudStack	1361	228	111

図1：テストコードに関する欠陥の件数が多かった
Apache Software Foundationのプロジェクト

テストコードにおける欠陥の発生頻度

- Apacheプロジェクトの不具合管理システムとバージョン管理システムを対象
 - * Apacheの計448のサブプロジェクト中, 211プロジェクト(**47%**)でテストに関する欠陥を発見
 - * プロダクトコードを含まない欠陥報告($(A \cup B) - C$)のうち, 5,556件のテストコードに係る欠陥報告を抽出
 - ▶ 手作業でランダムにサンプリングした443件の欠陥報告のうち, Silent Horrorは15件(**全体の3%**)



Silent Horror

- Silent Horrorは**Assertion Fault**と**Missing Assertion**に分けられる
 - * **Assertion Fault** : 誤ったアサーションがテストコードに含まれることによって発生するSilent Horrors
 - * **Missing Assertion** : アサーションの欠落によって発生するSilent Horrors

```
1  try {
2      nsp.unregisterNamespace("NotCurrentlyRegistered←
          ");
3  +   fail("Trying to unregister an unused prefix ←
          must fail");
4  } catch (NamespaceException e) {
5      // expected behaviour
6  }
```

アサーション : 前提として満たされるべき条件を記し,
満たされていない場合はエラーや例外を発生させる機能

図3 : Missing Assertionの例

欠陥JCR-500の修正コミット

(Jackrabbit Content Repository)

既存研究の問題点

-  • Silent Horror(偽陰性)を引き起こした欠陥の種類については調査されている
-  • **Silent Horror(偽陰性)が引き起こされた原因については明らかにしていない**

研究の目的

- 本研究ではソフトウェアテストの偽陰性の発生原因を明らかにすることを目指す
 - * しかしながら，偽陰性かどうかを特定するのは困難
- まず，テストコードの欠陥の発生原因の分析を行う

分析方法と リサーチクエスチョン

分析対象

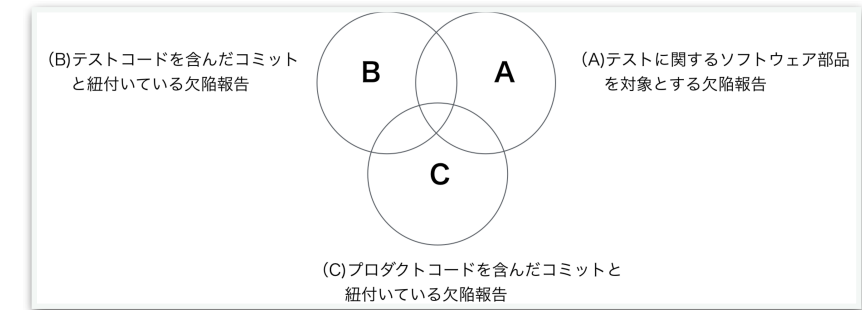


図2: 収集したデータ (既存研究の図を改変)

- 図2のAにあたるテストに関するソフトウェア部品を対象とする欠陥報告
 - * その中で、**CLOSED(修正済み)**されており、**且つ、REOPENED(再開)のタグ付けがされた欠陥報告**を分析対象とする
- REOPENEDを用いる理由
 - * テストコードを修正する必要があった原因とテストコードの修正パッチについての議論がある
 - * **テストコードの欠陥の発生原因を特定することが可能であるため**

リサーチクエストション

- REOPENEDされた欠陥に対するテストコードが偽陰性の発生原因であるかどうかを明らかにするために、以下のリサーチクエストションに取り組む
 - * **RQ1：REOPENEDされたテストコードの欠陥の発生原因は何か？**
 - * **RQ2：REOPENEDされたテストコードの欠陥はどの程度の頻度で偽陰性を引き起こすのか？**

対象プロジェクト



- **Apache Derby**

- * オープンソースのリレーショナルデータベース管理システム

- ▶ Javaで実装されている
 - ▶ APIとしての任意のJavaデータベースの結果に組み込みJDBC (Java Database Connectivity) ドライバを提供

- 既存研究[1]が分析したApache Software Foundationのプロジェクトの中で、**Apache Derbyプロジェクトが最もテストコードに関する欠陥の件数が多い**

- * プロダクトコードに対して記述されたテストコードの比率が最も多い

データの抽出

- Derbyのテストコードの欠陥報告は**Apache jira**[4]に記載
- タイプやステータスなどのタグによる絞り込み
 - * Type: **Bug** Status: **Closed** Component/s: **Test**
- テストコードの欠陥報告件数は**3496件**, このうち
REOPENEDにタグ付けされていたものは**70件(2%)**だった

目視による偽陰性の分析

- 70件の欠陥報告について，OPENからCLOSEDされるまでのテストコードの差分と欠陥報告に寄せられたコメントを目視する

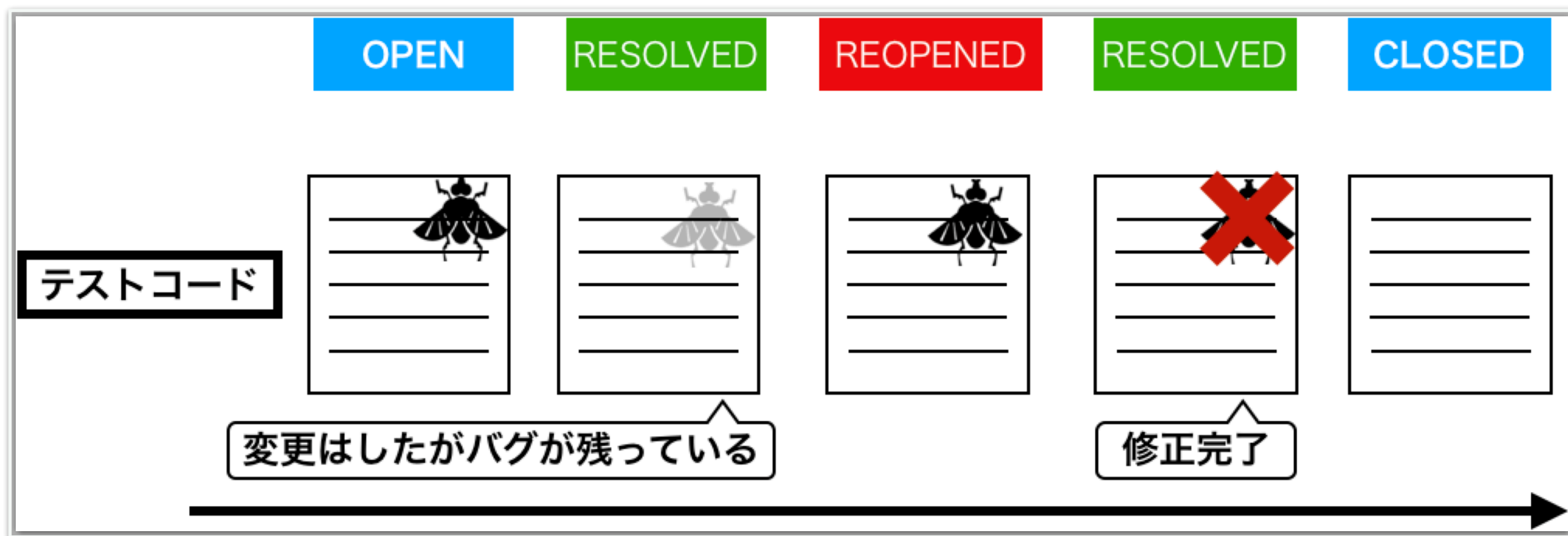


図4：テストコードの修正例

目視による分析

1. OPENからCLOSEDまでのコメントを目視

- * 発生した欠陥について理解

2. 報告された欠陥が発生した原因, 修正するために行った変更, および修正することができなかった変更について調査

3. テストコードの差分を目視

- * 実際にコメントで寄せられた修正が行われているか確認

偽陰性の可能性がある テストコードの欠陥例

ケース 1

- (問題) RecoveryAfterBackupTest というテストを実施した際に、断続的にアクセスが拒否されるという欠陥報告
 - * データベースが強制的にシャットダウンさせられる
- (原因) 既定のセキュリティ管理においてアクセス許可が必要ない場面であるにも関わらず ContextService.java において実装されていた
- (対策) アクセス認証を削除する

ケース 2

- (問題)SecureServerTestにおいて、サブプロセスからのstdout(標準出力)に対するアサーションエラーが発生するという欠陥報告
 - * サーバを起動する際に競合状態となることがあった
 - * テストがパスしている場合も存在していた
- (原因) 関数serverCameUpの呼び出しの”前”に、サーバの出力を収集したデータを格納する変数serverOutputに対してアサーションを表示するように記述されているが、直後にサーバの起動を待機するためのping/waitループが記述されており、この順番だと競合を起こす
- (対策)実際に起動したことを確認した”後”にサーバの出力を収集する
 - * ping/waitループの後に関数serverCameUpを移動させる

ケース 3

- (問題)J2EEDataSourceTestにおいて、例外処理の記述漏れが発生している箇所が複数発見されるという欠陥報告
 - * 予期しない例外がスローされた場合には失敗するが、try文のコードが何も例外をスローしない場合には失敗しないために見つけにくい
- (原因)**try文のコードが例外をスローしない場合には失敗しないため、エラーが発生せず見逃されてしまった**
- (対策)各try文内の最後にfail(“shutdown should raise exception”)を追加する
 - * このようなfail()の書き忘れやtry文におけるテストの欠陥は既存研究においても指摘されており、Derbyでもいくつか確認された

ケース 3 のコード

```
JDBCDataSource.setBeanProperty(ds, "shutdownDatabase", "shutdown");  
try {  
    ds.getConnection();  
+    fail("shutdown should raise exception");  
} catch (SQLException sqle) {  
    assertSQLState("XJ015", sqle);  
}
```

図 5 : J2EEDataSourceTestにおける欠陥と修正

図 3 と同様にfail()の記述漏れが原因

現状と今後の予定

70件の分析

- REOPENEDにタグ付けされた70件については一旦分析完了
 - * 不明な欠陥報告がある
 - ▶ コミットログが長すぎて時間がかかるため後回しにしたもの
 - 早急に分析を行う
 - ▶ JIRAに報告されたコミットが存在していない
 - 全く存在していないパターン
 - 存在はしているがJIRAの報告内容とテストコードが違うパターン

偽陰性の分類

- 偽陰性と思われるテストコードの欠陥を既存研究の **Assertion Fault**と**Missing Assertion**に分ける
 - * ただし既存研究ではアサーションに関する偽陰性を 2 つに分けただけ
 - * 2 つに当てはまらないものも存在するはず
 - ▶ アサーションに関するものではない偽陰性もある